General Computer Science
320201 GenCS I & II Problems

Michael Kohlhase

School of Engineering & Science
Jacobs University, Bremen Germany
m.kohlhase@jacobs-university.de

November 24, 2012

Preface

This document contains selected homework and self-study problems for the course General Computer Science I/II held at Jacobs University Bremen[1] in the academic years 2003-2012. It is meant as a supplement to the course notes [Gen11a, Gen11c]. We try to keep the numbering consistent between the documents.

This document contains practice and homework problems for the material coverd in the lecture (notes). The problems are tailored for understanding and practicing and should be attempted without consulting the solutions, which are avaialbe at [Gen11b, Gen11d]

This document is made available for the students of this course only. It is still a draft, and will develop over the course of the course. It will be developed further in coming academic years.

Acknowledgments: Immanuel Normann, Christoph Lange, Christine Müller, and Vyacheslav Zholudev have acted as lead teaching assistants for the course, have contributed many of the initial problems and organized them consistently. Throughout the time I have tought the course, the teaching assistants (most of them Jacobs University undergraduates; see below) have contributed new problems and sample solutions, have commented on existing problems and refined them.

GenCS Teaching Assistants: The following Jacobs University students have contributed problems while serving as teaching assiatants over the years: Darko Pesikan, Nikolaus Rath, Florian Rabe, Andrei Aiordachioaie, Dimitar Asenov, Alen Stojanov, Felix Schlesinger, Ştefan Anca, Anca Dragan, Vladislav Perelman, Josip Djolonga, Lucia Ambrošová, Flavia Grosan, Christoph Lange, Ankur Modi, Gordan Ristovski, Darko Makreshanski, Teodora Chitiboj, Cristina Stancu-Mara, Alin Iacob, Vladislav Perelman, Victor Savu, Mihai Cotizo Sima, Radu Cimpeanu, Mihai Cîlănaru, Maria Alexandra Alecu, Miroslava Georgieva Slavcheva, Corneliu-Claudiu Prodescu, Flavia Adelina Grosan, Felix Gabriel Mance, Anton Antonov, Alexandra Zayets, Ivaylo Enchev.

[1] International University Bremen until Fall 2006

Contents

0.1 Getting Started with "General Computer Science"

0.1.1 Overview over the Course

This should pose no problems

0.1.2 Administrativa

Neither should the administrativa

0.1.3 Motivation and Introduction

Problem 0.1 (Algorithms)
One of the most essential concepts in computer science is the Algorithm.

- What is the intuition behind the term "algorithm".
- What determines the quality of an algorithm?
- Give an everyday example of an algorithm.

Problem 0.2 (Keywords of General Computer Science)
Our course started with a motivation of "General Computer Science" where some fundamental notions where introduced. Name three of these fundamental notions and give for each of them a short explanation.

Problem 0.3 (Representations)
An essential concept in computer science is the Representation.

- What is the intuition behind the term "representation"?
- Why do we need representations?
- Give an everyday example of a representation.

0.2 Motivation and Introduction

Problem 0.4 (Algorithms)
One of the most essential concepts in computer science is the Algorithm.

- What is the intuition behind the term "algorithm".
- What determines the quality of an algorithm?
- Give an everyday example of an algorithm.

Problem 0.5 (Keywords of General Computer Science)
Our course started with a motivation of "General Computer Science" where some fundamental notions where introduced. Name three of these fundamental notions and give for each of them a short explanation.

Problem 0.6 (Representations)
An essential concept in computer science is the Representation.

- What is the intuition behind the term "representation"?
- Why do we need representations?
- Give an everyday example of a representation.

1 Representation and Computation

1.1 Elementary Discrete Math

1.1.1 Mathematical Foundations: Natural Numbers

25pt

Problem 1.1 (A wrong induction proof)
What is wrong with the following "proof by induction"?

> **Theorem:** All students of Jacobs University have the same hair color.
>
> **Proof:** We prove the assertion by induction over the number n of students at Jacobs University.
>
> **base case:** $n = 1$. If there is only one student at Jacobs University, then the assertion is obviously true.
>
> **step case:** $n > 1$. We assume that the assertion is true for all sets of n students and show that it holds for sets of $n+1$ students. So let us take a set S of $n+1$ students. As $n > 1$, we can choose students $s \in S$ and $t \in S$ with $s \neq t$ and consider sets $S_s = S \backslash \{s\}$ and $S_t := S \backslash \{t\}$. Clearly, $\#(S_s) = \#(S_t) = n$, so all students in S_s and have the same hair-color by inductive hypothesis, and the same holds for S_t. But $S = S_s \cup S_t$, so any $u \in S$ has the same hair color as the students in $S_s \cap S_t$, which have the same hair color as s and t, and thus all students in S have the same hair color □

Problem 1.2 (Natural numbers)
Prove or refute that $s(s(o))$ and $s(s(s(o)))$ are unary natural numbers and that their successors are different.

Problem 1.3 (Peano's induction axiom)
State Peano's induction axiom and discuss what it can be used for.

1.1.2 Naive Set Theory

25pt

Problem 1.4: Let A be a set with n elements (i.e $\#(A) = n$). What is the cardinality of the power set of A, (i.e. what is $\#(\mathcal{P}(A))$)?

15pt

Problem 1.5: Let $A := \{5, 23, 7, 17, 6\}$ and $B := \{3, 4, 8, 23\}$. Which of the relations are reflexive, antireflexive, symmetric, antisymmetric, and transitive?

Note: Please justify the answers.

$$
\begin{aligned}
R_1 \subseteq A \times A, R_1 &= \{\langle 23, 7\rangle, \langle 7, 23\rangle, \langle 5, 5\rangle, \langle 17, 6\rangle, \langle 6, 17\rangle\} \\
R_2 \subseteq B \times B, R_2 &= \{\langle 3, 3\rangle, \langle 3, 23\rangle, \langle 4, 4\rangle, \langle 8, 23\rangle, \langle 8, 8\rangle, \langle 3, 4\rangle, \langle 23, 23\rangle, \langle 4, 23\rangle\} \\
R_3 \subseteq B \times B, R_3 &= \{\langle 3, 3\rangle, \langle 3, 23\rangle, \langle 8, 3\rangle, \langle 4, 23\rangle, \langle 8, 4\rangle, \langle 23, 23\rangle\}
\end{aligned}
$$

20pt

Problem 1.6: Given two relations $R \subseteq C \times B$ and $Q \subseteq C \times A$, we define a relation $P \subseteq C \times (B \cap A)$ such that for every $x \in C$ and every $y \in (B \cap A)$, $\langle x, y\rangle \in P \Leftrightarrow \langle x, y\rangle \in R \vee \langle x, y\rangle \in Q$. Prove or refute (by giving a counterexample) the following statement: If Q and P are total functions, then P is a partial function.

1.1.3 Naive Set Theory

3pt
3min

Problem 1.7: Fill in the blanks in the table of Greek letters. Note that capitalized names denote capital Greek letters.

Symbol					γ	Σ	π	Φ
Name	alpha	eta	lambda	iota				

1.1.4 Relations and Functions

Problem 1.8 (Associativity of Relation Composition)

Let R, S, and T be relations on a set M. Prove or refute that the composition operation for relations is associative, i. e. that

$$(T \circ S) \circ R = T \circ (S \circ R)$$

1.2 Computing with Functions over Inductively Defined Sets

1.2.1 Standard ML: Functions as First-Class Objects

Problem 1.9: Define the `member` relation which checks whether an integer is member of a list of integers. The solution should be a function of type `int * int list -> bool`, which evaluates to `true` on arguments `n` and `l`, iff `n` is an element of the list `l`.

Problem 1.10: Define the `subset` relation. Set T is a subset of S iff all elements of T are also elements of S. The empty set is subset of any set.

Hint: Use the member function from Problem 1.9

20pt

Problem 1.11: Define functions to zip and unzip lists. zip will take two lists as input and create pairs of elements, one from each list, as follows: `zip [1,2,3] [0,2,4]` $\rightsquigarrow$ `[[1,0],[2,2],[3,4]]`. `unzip` is the inverse function, taking one list of tuples as argument and outputing two separate lists. `unzip [[1,4],[2,5],[3,6]]` $\rightsquigarrow$ `[1,2,3] [4,5,6]`.

Problem 1.12 (Compressing binary lists)
Define a data type of binary digits. Write a function that takes a list of binary digits and returns an int list that is a compressed version of it and the first binary digit of the list (needed for reconversion). For example,

```
ZIPit([zero,zero,zero, one,one,one,one,
 zero,zero,zero, one, zero,zero]) -> (0,[3,4,3,1,2]),
```

because the binary list begins with 3 zeros, followed by 4 ones etc.

Problem 1.13 (Decompressing binary lists)
Write an inverse function `UNZIPit` of the one written in Problem 1.12. 15pt

Problem 1.14: Program the function f with $f(x) = x^2$ on unary natural numbers without using the multiplication function.

20pt

Problem 1.15 (Translating between Integers and Strings)
SML has pre-defined types `int` and `string`, write two conversion functions:

- `int2string` converts an integer to a string, i.e. `int2string(~317)` $\rightsquigarrow$ `"~317":string`
- `string2int` converts a suitable string to an integer, i.e. `string2int("444")` $\rightsquigarrow$ `444:int`. For the moment, we do not care what happens, if the input string is unsuitable, i.e does not correspond to an integer.

do not use any built-in functions except elementary arithmetic (which include `mod` and `div` BTW), `explode`, and `implode`.

Problem 1.16: Write a function that takes an odd positive integer and returns a `char list list` which represents a triangle of stars with n stars in the last row. For example,

```
triangle 5;
val it =
[#" ", #" ", #"*", #" ", #" "],
[#" ", #"*", #"*", #"*", #" "],
[#"*", #"*", #"*", #"*", #"*"]]
```

Problem 1.17: Write a non-recursive variant of the `member` function from Problem 1.9 using the `foldl` function.

15pt

Problem 1.18 (Decimal representations as lists) 10min
The decimal representation of a natural number is the list of its digits (i. e. integers between 0 and 9). Write an SML function `decToInt` of type `int list -> int` that converts the decimal representation of a natural number to the corresponding number:

```
- decToInt [7,8,5,6];
val it = 7856 : int
```

Hint: Use a suitable built-in higher-order list function of type `fn : (int * int -> int) -> int -> int list -> int` that solves a great part of the problem.

30pt

Problem 1.19 (List functions via `foldl`/`foldr`)
Write the following procedures using `foldl` or `foldr`

1. `length` which computes the length of a list
2. `concat`, which gets a list of lists and concatenates them to a list.
3. `map`, which maps a function over a list
4. `myfilter`, `myexists`, and `myforall` from ??

10pt

Problem 1.20 (Mapping and Appending)
Can the functions `mapcan` and `mapcan2` be written using `foldl`/`foldr`?

1.2.2 Inductively Defined Sets and Computation

Problem 1.21: Figure out the functions on natural numbers for the following defining equations

$$\tau(o) = o$$

$$\tau(s(n)) = s(s(s(\tau(n))))$$

15pt
5min

Problem 1.22 (A function on natural numbers)
Figure out the function on natural numbers defined by the following equations:

$$\eta(o) = o$$

$$\eta(s(o)) = o$$

$$\eta(s(s(n))) = s(\eta(n))$$

15pt

Problem 1.23: In class, we have been playing with defining equations for functions on the natural numbers. Give the defining equations for the function σ with $\sigma(x) = x^2$ without using the multiplication function (you may use the addition function though). Prove from the Peano axioms or refute by a counterexample that your equations define a function. Indicate in each step which of the axioms you have used.

1.2.3 Inductively Defined Sets in SML

Problem 1.24: Declare an SML datatype `pair` representing pairs of integers and define SML functions `fst` and `snd` where `fst` returns the first- and `snd` the second component of q the pair. Moreover write down the type of the constructor of `pair` as well as of the two procedures `fst` and `snd`. 8pt 8min

Use SML syntax for the whole problem.

Problem 1.25: Declare a data type `myNat` for unary natural numbers and `NatList` for lists of natural numbers in SML syntax, and define a function that computes the length of a list (as a unary natural number in `mynat`). Furthermore, define a function `nms` that takes two unary natural numbers `n` and `m` and generates a list of length `n` which contains only `m`s, i.e. `nms(s(s(zero)),s(zero))` evaluates to `construct(s(zero),construct(s(zero),elist))`. 4pt 8min

Problem 1.26: Given the following SML data type for an arithmetic expressions 20pt

```
datatype arithexp = aec of int (* 0,1,2,... *)
                  | aeadd of arithexp * arithexp (* addition *)
                  | aemul of arithexp * arithexp (* multiplication *)
                  | aesub of arithexp * arithexp (* subtraction *)
                  | aediv of arithexp * arithexp (* division *)
                  | aemod of arithexp * arithexp (* modulo *)
                  | aev of int (* variable *)
```

give the representation of the expression $(4x + 5) - 3x$.

Write a (cascading) function `eval : (int -> int) -> arithexp -> int` that takes a variable assignment φ and an arithmetic expresson e and returns its evaluation as a value.

Note: A variable assignment is a function that maps variables to (integer) values, here it is represented as function φ of type `int -> int` that assigns $\varphi(n)$ to the variable `aev(`n`)`.

Problem 1.27 (Your own lists)
Define a data type `mylist` of lists of integers with constructors `mycons` and `mynil`. Write translators `tosml` and `tomy` to and from SML lists, respectively.

Problem 1.28 (Unary natural numbers)
Define a `datatype nat` of unary natural numbers and implement the functions

- `add = fn : nat * nat -> nat` (adds two numbers)
- `mul = fn : nat * nat -> nat` (multiplies two numbers)

Problem 1.29 (Nary Multiplication)
By defining a new datatype for n-tuples of unary natural numbers, implement an n-ary multiplications using the function `mul` from Problem 1.28. For $n = 1$, an n-tuple should be constructed by using a constructor named `first`; for $n > 1$, further elements should be prepended to the first by using a constructor named `next`. The multiplication function `nmul` should return the product of all elements of a given tuple.

For example,

```
nmul(next(s(s(zero)),
       next(s(s(zero)),
         first(s(s(s(zero)))))))
```

should output `s(s(s(s(s(s(s(s(s(s(s(s(zero))))))))))))` since 223 = 12.

1.2.4 A Theory of SML: Abstract Data Types and Term Languages

5pt
Abstract Data Types and Ground Constructor Terms

Problem 1.30: Translate the abstract data types given in mathematical notation into SML datatypes

1. $\langle\{\mathbb{S}\}, \{[c_1 \colon \mathbb{S}], [c_2 \colon \mathbb{S} \to \mathbb{S}], [c_3 \colon \mathbb{S} \times \mathbb{S} \to \mathbb{S}], [c_4 \colon \mathbb{S} \to \mathbb{S} \to \mathbb{S}]\}\rangle$
2. $\langle\{\mathbb{T}\}, \{[c_1 \colon \mathbb{T}], [c_2 \colon \mathbb{T} \times (\mathbb{T} \to \mathbb{T}) \to \mathbb{T}]\}\rangle$

5pt
5min

Problem 1.31: Translate the given SML datatype

5min

```
datatype T = 0 | c1 of T * T | c2 of T -> (T * T)
```

into abstract data type in mathmatical notation.

20pt

Problem 1.32 (Nested lists)
In class, we have defined an abstract data type for lists of natural numbers. Using this intuition, construct an abstract data type for lists that contain natural numbers or lists (nested up to arbitrary depth). Give the constructor term (the trace of the construction rules) for the list $[3, 4, [7, [8, 2], 9], 122, [2, 2]]$.

A First Abstract Interpreter **Problem 1.33:** Give the defining equations for the maximum function for two numbers. This function takes two arguments and returns the larger one. 30pt

Hint: You may define auxiliary functions with defining equations of their own. You can use ι from above.

Problem 1.34: Using the abstract data type of truth functions from ??, give the defining equations for a function ι that takes three arguments, such that $\iota(\varphi_{\mathbb{B}}, a_{\mathbb{N}}, b_{\mathbb{N}})$ behaves like "if φ then a, else b", where a and b are natural numbers. 15pt

Problem 1.35: Consider the following abstract data type: 6pt

$$\mathcal{A} := \langle\{\mathbb{A}, \mathbb{B}, \mathbb{C}\}, \{[f\colon \mathbb{C} \to \mathbb{B}], [g\colon \mathbb{A} \times \mathbb{B} \to \mathbb{C}], [h\colon \mathbb{C} \to \mathbb{A}], [a\colon \mathbb{A}], [b\colon \mathbb{B}], [c\colon \mathbb{C}]\}\rangle$$

Which of the following expressions are constructor terms (with variables), which ones are ground. Give the sorts for the terms.

Answer with **Y**es or **N**o or /. and give the sort (if term)			
expression	term?	ground?	Sort
$f(g(a))$			
$f(g(\langle a, b\rangle))$			
$h(g(\langle h(x_{\mathbb{C}}), f(c)\rangle))$			
$h(g(\langle h(x_{\mathbb{B}}), f(y_{\mathbb{C}})\rangle))$			

Problem 1.36 (Substitution) Substitutions 5pt

Apply the substitutions $\sigma := [b/x], [(g(a))/y], [a/w]$ and $\tau := [(h(c))/x], [c/z]$ to the terms $s := f(g(x, g(a, x, b), y))$ and $t := g(x, x, h(y))$ (give the 4 result terms $\sigma(s)$, $\sigma(t)$, $\tau(s)$, and $\tau(t)$). 5min

Definition 1 We call a substitution σ **idempotent**, iff $\sigma(\sigma(\mathbf{A})) = \sigma(\mathbf{A})$ for all terms $\mathbf{A}$.

Definition 2 For a substitution $\sigma = [\mathbf{A}_1/x_1], \cdots, [\mathbf{A}_n/x_n]$, we call the set $\mathbf{intro}(\sigma) := \bigcup_{1 \leq i \leq n} \mathbf{free}(\mathbf{A}_i)$ the set of variables **introduced** by σ, and the set $\mathbf{supp}(\sigma) := \{x_i \mid 1 \leq i \leq n\}$

30pt

Problem 1.37: Prove or refute that σ is idempotent, if $\mathbf{intro}(\sigma) \cap \mathbf{supp}(\sigma) = \emptyset$.

Problem 1.38 (Substitution Application) 30pt

Consider the following SML data type of terms:

```
datatype term = const of string
              | var of string
              | pair of term * term
              | appl of string * term
```

Constants and variables are represented by a constructor taking their name string, whereas applications of the form $f(t)$ are constructed from the name string and the argument. Remember that we use $f(a, b)$ as an abbreviation for $f(\langle a, b\rangle)$. Thus a term $f(a, g(x))$ is represented as `appl("f",pair(const("a"), appl("g", var("x"))))`.

With this, we can represent substitutions as lists of elementary substitutions, which are pairs of type `term * string`. Thus we can set

```
type subst = term * string list
```

and represent a substitution $\sigma = [(f(a))/x], [b/y]$ as `[(appl("f", const("a")), "x"), (const("b"), "y")]`. Of course we may not allow ambiguous substitutions which contain duplicate strings.

Write an SML function `substApply` for the substitution application operation, i.e. `substApply` takes a substitution σ and a term $\mathbf{A}$ as arguments and returns the term $\sigma(\mathbf{A})$ if σ is unambiguous and raises an exception otherwise.

Make sure that your function applies substitutions in a parallel way, i.e. that $[y/x], [x/z](f(z)) = f(x)$.

A Second Abstract Interpreter **Problem 1.39:** Consider the following abstract procedure on the abstract data type of natural numbers: 20pt

$$\mathcal{P} := \langle f{::}\mathbb{N} \to \mathbb{N}\,;\ \{f(o) \rightsquigarrow o, f(s(o)) \rightsquigarrow o, f(s(s(n_{\mathbb{N}}))) \rightsquigarrow s(f(n_{\mathbb{N}}))\}\rangle$$

1. Show the computation process for $\mathcal{P}$ on the arguments $s(s(s(o)))$ and $s(s(s(s(s(s(o))))))$.
2. Give the recursion relation of $\mathcal{P}$.
3. Does $\mathcal{P}$ terminate on all inputs?
4. What function is computed by $\mathcal{P}$?

Evaluation Order and Termination

Problem 1.40: Explain the concept of a "call-by-value" programming language in terms of evaluation order. Give an example program where this effects evaluation and termination, explain it. 4pt 10min

---

Note: One point each for the definition, the program and the explanation.

Problem 1.41: Give an example of an abstract procedure that diverges on all arguments, and another one that terminates on some and diverges on others, each example with a short explanation. 2pt 5min

Problem 1.42: Give the recursion relation of the abstract procedures in Problem 1.14, ??, ??, and Problem 1.56 and discuss termination. 15pt

1.2.5 More SML: Recursion in the Real World

No problems supplied yet.

1.2.6 Even more SML: Exceptions and State in SML

Problem 1.43 (Integer Intervals) 5pt 10min

Declare an SML data type for natural numbers and one for lists of natural numbers in SML. Write an SML function that given two natural number n and m (as a constructor term) creates the list `[n,n+1,\ldots,m-1,m]` if $n \leq m$ and raises an exception otherwise.

Problem 1.44 (Operations with Exceptions)

Add to the functions from Problem 1.28 functions for subtraction and division that raise exceptions where necessary.

- `function sub: nat*nat -> nat` (subtracts two numbers)
- `function div: nat*nat -> nat` (divides two numbers)

Problem 1.45 (List Functions with Exceptions) 6pt 20min

Write three SML functions `nth`, `take`, `drop` that take a list and an integer as arguments, such that

1. `nth(xs,n)` gives the `n`-th element of the list `xs`.
2. `take(xs,n)` returns the list of the first `n` elements of the list `xs`.
3. `drop(xs,n)` returns the list that is obtained from `xs` by deleting the first `n` elements.

In all cases, the functions should raise the exception `Subscript`, if $n < 0$ or the list `xs` has less than `n` elements. We assume that list elements are numbered beginning with 0.

Problem 1.46 (Transformations with Errors) 10pt

Extend the function from Problem 1.15 by an error flag, i.e. the value of the function should be a pair consisting of a string, and the boolean value `true`, if the string was suitable, and `false` if it was not.

Problem 1.47 (Simple SML data conversion) 10pt

Write an SML function `char_to_int = fn : char -> int` that given a single character in the range $[0-9]$ returns the corresponding integer. Do not use the built-in function `Int.fromString` but do the character parsing yourself. If the supplied character does not represent a valid digit raise an `InvalidDigit` exception. The exception should have one parameter that contains the invalid character, i.e. it is defined as `exception InvalidDigit of char`

Problem 1.48 (Strings and numbers) 10pt

Write two SML functions

1. `str_to_int = fn : string -> int`
2. `str_to_real = fn : string -> real`

that given a string convert it to an integer or a real respectively. Do not use the built-in functions `Int.fromString, Real.fromString` but do the string parsing yourself.

- Negative numbers begin with a `'~'` character (not '-').
- If the string does not represent a valid integer raise an exception as in the previous exercise. Use the same definition and indicate which character is invalid.
- If the input string is empty raise an exception.
- Examples of valid inputs for the second function are: `~1, ~1.5, 4.63, 0.0, 0, .123`

Problem 1.49 (Recursive evaluation) 10pt

Write an SML function `evaluate = fn : expression -> real` that takes an expression of the following datatype and computes its value:

```
datatype expression = add of expression*expression (* add *)
                    | sub of expression*expression (* subtract *)
                    | dvd of expression*expression (* divide *)
                    | mul of expression*expression (* multiply *)
                    | num of real;
```

For example we have

```
evaluate(num(1.3)) -> 1.3
evaluate(div(num(2.2),num(1.0))) -> 2.2
evaluate(add(num(4.2),sub(mul(num(2.1),num(2.0)),num(1.4)))) -> 7.0
```

10pt

Problem 1.50 (List evaluation)
Write a new function `evaluate_list = fn : expression list -> real list` that evaluates a list of expressions and returns a list with the corresponding results. Extend the `expression` datatype from the previous exercise by the additional constructor: `var of int`.

The variables here are the final results of previously evaluated expressions. I.e. the first expression from the list should not contain any variables. The second can contain the term `var(0)` which should evaluate to the result from the first expression and so on . . . If an expression contains an invalid variable term raise: `exception InvalidVariable of int` that indicates what identifier was used for the variable.

For example we have

```
evaluate_list [num(3.0), num(2.5), mul(var(0),var(1))] -> [3.0,2.5,7.5]
```

10pt

Problem 1.51 (String parsing)
Write an SML function `evaluate_str = fn : string list -> real list` that given a list of arithmetic expressions represented as strings returns their values. The strings follow the following conventions:

- strict bracketing: every expression consists of 2 operands joined by an operator and has to be enclosed in brackets, i.e. $1 + 2 + 3$ would be represented as `((1+2)+3)` (or `(1+(2+3))`)
- no spaces: the string contains no empty characters

The value of each of the expressions is stored in a variable named vn with n the position of the expression in the list. These variables can be used in subsequent expressions.

Raise an exception `InvalidSyntax` if any of the strings does not follow the conventions.

For example we have

```
evaluate_str ["((4*.5)-(1+2.5))"] -> [~1.5]
evaluate_str ["((4*.5)-(1+2.5))","(v0*~2)"] -> [~1.5,3.0]
evaluate_str ["(1.8/2)","(1-~3)","(v0+v1)"] -> [0.9,4.0,4.9]
```

10pt

Problem 1.52 (SML File IO)
Write an SML function `evaluate_file = fn : string -> string -> unit` that performs file IO operations. The first argument is an input file name and the second is an output file name. The input file contains lines which are arithmetic expressions. `evaluate_file` reads all the expressions, evaluates them, and writes the corresponding results to the output file, one result per line.

For example we have

```
evaluate_list "input.txt" "output.txt";

Contents of input.txt:
4.9
0.7
(v0/v1)

Contents of output.txt (after evaluate_list is executed):
4.9
0.7
7.0
```

1.3 A Theory of SML: Abstract Data Types and Term Languages

1.3.1 Abstract Data Types and Ground Constructor Terms

Problem 1.53: Translate the abstract data types given in mathematical notation into SML datatypes 5pt 5min

1. $\langle\{\mathbb{S}\},\{[c_1\colon\mathbb{S}],[c_2\colon\mathbb{S}\to\mathbb{S}],[c_3\colon\mathbb{S}\times\mathbb{S}\to\mathbb{S}],[c_4\colon\mathbb{S}\to\mathbb{S}\to\mathbb{S}]\}\rangle$
2. $\langle\{\mathbb{T}\},\{[c_1\colon\mathbb{T}],[c_2\colon\mathbb{T}\times(\mathbb{T}\to\mathbb{T})\to\mathbb{T}]\}\rangle$

Problem 1.54: Translate the given SML datatype 5pt 5min

```
datatype T = 0 | c1 of T * T | c2 of T -> (T * T)
```

into abstract data type in mathmatical notation.

Problem 1.55 (Nested lists) 20pt

In class, we have defined an abstract data type for lists of natural numbers. Using this intuition, construct an abstract data type for lists that contain natural numbers or lists (nested up to arbitrary depth). Give the constructor term (the trace of the construction rules) for the list $[3,4,[7,[8,2],9],122,[2,2]]$.

1.3.2 A First Abstract Interpreter

Problem 1.56: Give the defining equations for the maximum function for two numbers. This function takes two arguments and returns the larger one. 30pt

---

Hint: You may define auxiliary functions with defining equations of their own. You can use ι from above.

Problem 1.57: Using the abstract data type of truth functions from ??, give the defining equations for a function ι that takes three arguments, such that $\iota(\varphi_{\mathbb{B}}, a_{\mathbb{N}}, b_{\mathbb{N}})$ behaves like "if φ then a, else b", where a and b are natural numbers. 15pt

Problem 1.58: Consider the following abstract data type: 6pt

$$\mathcal{A} := \langle \{\mathbb{A}, \mathbb{B}, \mathbb{C}\}, \{[f\colon \mathbb{C} \to \mathbb{B}], [g\colon \mathbb{A} \times \mathbb{B} \to \mathbb{C}], [h\colon \mathbb{C} \to \mathbb{A}], [a\colon \mathbb{A}], [b\colon \mathbb{B}], [c\colon \mathbb{C}]\}\rangle$$

Which of the following expressions are constructor terms (with variables), which ones are ground. Give the sorts for the terms.

Answer with **Y**es or **N**o or /. and give the sort (if term)			
expression	term?	ground?	Sort
$f(g(a))$			
$f(g(\langle a, b\rangle))$			
$h(g(\langle h(x_{\mathbb{C}}), f(c)\rangle))$			
$h(g(\langle h(x_{\mathbb{B}}), f(y_{\mathbb{C}})\rangle))$			

1.3.3 Substitutions

Problem 1.59 (Substitution) 4pt 5min

Apply the substitutions $\sigma := [b/x], [(g(a))/y], [a/w]$ and $\tau := [(h(c))/x], [c/z]$ to the terms $s := f(g(x, g(a, x, b), y))$ and $t := g(x, x, h(y))$ (give the 4 result terms $\sigma(s)$, $\sigma(t)$, $\tau(s)$, and $\tau(t)$).

Definition 3 We call a substitution σ **idempotent**, iff $\sigma(\sigma(\mathbf{A})) = \sigma(\mathbf{A})$ for all terms $\mathbf{A}$.

Definition 4 For a substitution $\sigma = [\mathbf{A}_1/x_1], \cdots, [\mathbf{A}_n/x_n]$, we call the set $\mathbf{intro}(\sigma) := \bigcup_{1 \leq i \leq n} \mathbf{free}(\mathbf{A}_i)$ the set of variables **introduced** by σ, and the set $\mathbf{supp}(\sigma) := \{x_i \mid 1 \leq i \leq n\}$

Problem 1.60: Prove or refute that σ is idempotent, if $\mathbf{intro}(\sigma) \cap \mathbf{supp}(\sigma) = \emptyset$. 30pt

Problem 1.61 (Substitution Application) 30pt

Consider the following SML data type of terms:

```
datatype term = const of string
              | var of string
              | pair of term * term
              | appl of string * term
```

Constants and variables are represented by a constructor taking their name string, whereas applications of the form $f(t)$ are constructed from the name string and the argument. Remember that we use $f(a, b)$ as an abbreviation for $f(\langle a, b\rangle)$. Thus a term $f(a, g(x))$ is represented as `appl("f",pair(const("a"), appl("g", var("x"))))`.

With this, we can represent substitutions as lists of elementary substitutions, which are pairs of type `term * string`. Thus we can set

```
type subst = term * string list
```

and represent a substitution $\sigma = [(f(a))/x], [b/y]$ as `[(appl("f", const("a")), "x"), (const("b"), "y")]`. Of course we may not allow ambiguous substitutions which contain duplicate strings.

Write an SML function `substApply` for the substitution application operation, i.e. `substApply` takes a substitution σ and a term $\mathbf{A}$ as arguments and returns the term $\sigma(\mathbf{A})$ if σ is unambiguous and raises an exception otherwise.

Make sure that your function applies substitutions in a parallel way, i.e. that $[y/x], [x/z](f(z)) = f(x)$.

1.3.4 A Second Abstract Interpreter

20pt

Problem 1.62: Consider the following abstract procedure on the abstract data type of natural numbers:

$$\mathcal{P} := \langle f{::}\mathbb{N} \to \mathbb{N}\,;\,\{f(o) \rightsquigarrow o, f(s(o)) \rightsquigarrow o, f(s(s(n_{\mathbb{N}}))) \rightsquigarrow s(f(n_{\mathbb{N}}))\}\rangle$$

1. Show the computation process for $\mathcal{P}$ on the arguments $s(s(s(o)))$ and $s(s(s(s(s(s(o))))))$.
2. Give the recursion relation of $\mathcal{P}$.
3. Does $\mathcal{P}$ terminate on all inputs?
4. What function is computed by $\mathcal{P}$?

1.3.5 Evaluation Order and Termination

Problem 1.63: Explain the concept of a "call-by-value" programming language in terms of evaluation order. Give an example program where this effects evaluation and termination, explain it. 4pt 10min

---

Note: One point each for the definition, the program and the explanation.

Problem 1.64: Give an example of an abstract procedure that diverges on all arguments, and another one that terminates on some and diverges on others, each example with a short explanation. 2pt 5min

Problem 1.65: Give the recursion relation of the abstract procedures in Problem 1.14, ??, ??, and Problem 1.56 and discuss termination. 15pt

1.4 More SML

1.4.1 More SML: Recursion in the Real World

No problems supplied yet.

1.4.2 Programming with Effects: Imperative Features in SML

Input and Output nothing here yet.

5pt

Problem 1.66 (Integer Intervals) **Even more SML: Exceptions and State in SML**

Declare an SML data type for natural numbers and one for lists of natural numbers in SML. Write an SML function that given two natural number n and m (as a constructor term) creates the list `[n,n+1,\ldots,m-1,m]` if $n \leq m$ and raises an exception otherwise.

Problem 1.67 (Operations with Exceptions) 10min

Add to the functions from Problem 1.28 functions for subtraction and division that raise exceptions where necessary.

- `function sub: nat*nat -> nat` (subtracts two numbers)
- `function div: nat*nat -> nat` (divides two numbers)

6pt

Problem 1.68 (List Functions with Exceptions) 20min

Write three SML functions `nth`, `take`, `drop` that take a list and an integer as arguments, such that

1. `nth(xs,n)` gives the `n`-th element of the list `xs`.
2. `take(xs,n)` returns the list of the first `n` elements of the list `xs`.
3. `drop(xs,n)` returns the list that is obtained from `xs` by deleting the first `n` elements.

In all cases, the functions should raise the exception `Subscript`, if $n < 0$ or the list `xs` has less than `n` elements. We assume that list elements are numbered beginning with 0.

10pt

Problem 1.69 (Transformations with Errors)

Extend the function from Problem 1.15 by an error flag, i.e. the value of the function should be a pair consisting of a string, and the boolean value `true`, if the string was suitable, and `false` if it was not.

10pt

Problem 1.70 (Simple SML data conversion)

Write an SML function `char_to_int = fn : char -> int` that given a single character in the range $[0-9]$ returns the corresponding integer. Do not use the built-in function `Int.fromString` but do the character parsing yourself. If the supplied character does not represent a valid digit raise an `InvalidDigit` exception. The exception should have one parameter that contains the invalid character, i.e. it is defined as `exception InvalidDigit of char`

10pt

Problem 1.71 (Strings and numbers)

Write two SML functions

1. `str_to_int = fn : string -> int`
2. `str_to_real = fn : string -> real`

that given a string convert it to an integer or a real respectively. Do not use the built-in functions `Int.fromString, Real.fromString` but do the string parsing yourself.

- Negative numbers begin with a `'~'` character (not '-').
- If the string does not represent a valid integer raise an exception as in the previous exercise. Use the same definition and indicate which character is invalid.

- If the input string is empty raise an exception.
- Examples of valid inputs for the second function are: `~1, ~1.5, 4.63, 0.0, 0, .123`

10pt

Problem 1.72 (Recursive evaluation)
Write an SML function `evaluate = fn : expression -> real` that takes an expression of the following datatype and computes its value:

```
datatype expression = add of expression*expression (* add *)
                    | sub of expression*expression (* subtract *)
                    | dvd of expression*expression (* divide *)
                    | mul of expression*expression (* multiply *)
                    | num of real;
```

For example we have

```
evaluate(num(1.3)) -> 1.3
evaluate(div(num(2.2),num(1.0))) -> 2.2
evaluate(add(num(4.2),sub(mul(num(2.1),num(2.0)),num(1.4)))) -> 7.0
```

10pt

Problem 1.73 (List evaluation)
Write a new function `evaluate_list = fn : expression list -> real list` that evaluates a list of expressions and returns a list with the corresponding results. Extend the `expression` datatype from the previous exercise by the additional constructor: `var of int`.

The variables here are the final results of previosly evaluated expressions. I.e. the first expression from the list should not contain any variables. The second can contain the term `var(0)` which should evaluate to the result from the first expression and so on . . . If an expression contains an invalid variable term raise: `exception InvalidVariable of int` that indicates what identifier was used for the variable.

For example we have

```
evaluate_list [num(3.0), num(2.5), mul(var(0),var(1))] -> [3.0,2.5,7.5]
```

10pt

Problem 1.74 (String parsing)
Write an SML function `evaluate_str = fn : string list -> real list` that given a list of arithmetic expressions represented as strings returns their values. The strings follow the following conventions:

- strict bracketing: every expression consists of 2 operands joined by an operator and has to be enclosed in brackets, i.e. $1 + 2 + 3$ would be represented as `((1+2)+3)` (or `(1+(2+3))`)
- no spaces: the string contains no empty characters

The value of each of the expressions is stored in a variable named vn with n the position of the expression in the list. These variables can be used in subsequent expressions.

Raise an exception `InvalidSyntax` if any of the strings does not follow the conventions.

For example we have

```
evaluate_str ["((4*.5)-(1+2.5))"] -> [~1.5]
evaluate_str ["((4*.5)-(1+2.5))","(v0*~2)"] -> [~1.5,3.0]
evaluate_str ["(1.8/2)","(1-~3)","(v0+v1)"] -> [0.9,4.0,4.9]
```

10pt

Problem 1.75 (SML File IO)
Write an SML function `evaluate_file = fn : string -> string -> unit` that performs file IO operations. The first argument is an input file name and the second is an output file name. The input file contains lines which are arithmetic expressions. `evaluate_file` reads all the expressions, evaluates them, and writes the corresponding results to the output file, one result per line.

For example we have

```
evaluate_list "input.txt" "output.txt";

Contents of input.txt:
4.9
0.7
(v0/v1)

Contents of output.txt (after evaluate_list is executed):
4.9
0.7
7.0
```

1.5 Encoding Programs as Strings

1.5.1 Formal Languages

Problem 1.76: Given the alphabet $A = \{a, b, c\}$ and a $L := \bigcup_{i=1}^{\infty} L_i$, where $L_1 = \{\epsilon\}$ and L_{i+1} contains the strings x, bbx, xac for all $x \in L_i$. 3pt 5min

1. Is L a formal language?
2. Which of the following strings are in L? Justify your answer

$s_1 = bbac$	$s_2 = bbacc$	$s_3 = bbbac$
$s_4 = acac$	$s_5 = bbbacac$	$s_6 = bbacac$

Problem 1.77: Given the alphabet $A = \{a, 2, \S\}$. 2pt

1. Determine $k = \#(Q)$ with $Q = \{s \in A^+ \mid |s| \leq 5\}$.
2. Is Q a formal language over A? Justify your results.

Problem 1.78: Let $A := \{\mathtt{a}, \mathtt{h}, /, \#, \mathtt{x}\}$ and $\prec$ be the ordering relation on A with $\mathtt{x} \prec \# \prec / \prec \mathtt{h} \prec \mathtt{a}$. Order the following strings in A^* in the lexical order $<_{\text{lex}}$ induced by $\prec$. 3pt 5min

$s_1 = \#\#\#\#$	$s_2 = \#\#\mathtt{x}\#\#\mathtt{h}$	$s_3 = \epsilon$
$s_4 = \#\#\mathtt{h}\#\#\mathtt{x}$	$s_5 = \mathtt{a}\#\#\#\mathtt{a}\#$	$s_6 = \#\#\#\#/$

Problem 1.79 (Lexical Ordering) 20pt

Write a lexical ordering function `lex` on lists in SML, such that `lex` takes three arguments, an ordering relation (i.e. a binary function from list elements to Booleans), and two lists (representing strings over an arbitrary alphabet). Then `lex(o,l,r)` compares lists `l` and `r` in the lexical ordering induced by the character ordering `o`.

We want the function `lex` to return three value strings `"l<r"`, `"r<l"`, and `"l=r"` with the obvious meanings.

1.5.2 Elementary Codes

2pt

Problem 1.80: Given the alphabets $A = \{\mathtt{a}, 2\}$ and $B = \{\mathtt{9}, \#, /\}$.

1. Is c with $c(a) = \#\#$ and $c(2) = \mathtt{9}\#\#\#/$ a character code?
2. Is the extension of c on strings over A a code?

30pt

Problem 1.81 (Testing for prefix codes)
Write an SML function `prefix_code` that tests whether a code is a prefix code. The code is given as a list of pairs (SML type `char*string list`).

Example:

```
prefix_code [(#"a","0"), (#"b","1")];
val it = true : bool
```

Hint: You have to test for functionhood, injectivity and the prefix property.

8pt
5min

Problem 1.82: Let $A := \{a, b, c, d, e, f, g, h\}$ and $\mathbb{B} := \{0, 1\}$, and

$c(a)$:=010010010101001	$c(b)$:=010110010101001
$c(c)$:=010011110101001	$c(d)$:=010010011101001
$c(e)$:=010010010110001	$c(f)$:=010010010101101
$c(g)$:=010011110101000	$c(h)$:=011111110101000

Is c a character code? Does it induce a code on strings?

40pt

Problem 1.83 (Morse Code Translator)
Write an SML program that transforms arbitrary strings into Morse Code. Write a translation function from Morse code to regular strings and show on some examples that the translators are inverses.

Hint: The Morse codes are multi-character strings. In the Morse representation of the string, these codes should be separated by space characters. This makes a back-translation possible.

20pt

Problem 1.84 (Morse Code again)
With what you know about codes now, is the Morse Code (without the blank characters as stop symbols) a code on strings? Give a proof for your answer.

30pt

Problem 1.85 (String Decoder without Stop Characters)
Write a general string decoder that takes as the first argument a code (in the representation you developed in Problem 1.81) and decodes strings with respect to this code if possible and raises and exception otherwise.

1.5.3 Character Codes in the Real World

No problems supplied yet.

1.5.4 Formal Languages and Meaning

No problems supplied yet.

1.6 Boolean Algebra

1.6.1 Boolean Expressions and their Meaning

15pt

Problem 1.86 (Boolean complements)
Prove or refute that the following is a theorem of Boolean Algebra:

> For all $a, b \in \mathbb{B}$, if both $a + b = 1$ and $a * b = 0$, we obtain $b = \overline{a}$. (That is, any $b \in \mathbb{B}$ has a unique complement, regardless of whether we're considering Boolean sums or products.)

Observation: You are not allowed to use truth tables in this proof. Give a solution that is only based on Boolean Algebra rules and theorems.

10pt

Problem 1.87: Give a model for C_{bool}, where the following expression are theorems: $a*\overline{a}$, $a+\overline{a}$, $a*a$, $\overline{a+a}$.

Hint: Give the truth tables for the Boolean functions.

15pt

Problem 1.88 (Partial orders in a Boolean algebra)
For a given boolean algebra with a universe $\mathbb{B}$ and $a, b \in \mathbb{B}$, we define that the relation $a \leq b$ holds iff $a + b = b$. Prove for refute that $\leq$ is a partial order on $\mathbb{B}$.

Note: There are boolean algebras with a universe $\mathbb{B}$ larger than just $\{0, 1\}$. We are not going to consider them in the scope of this lecture, but still try to keep your proof as generic as possible. That is, assume that a, b are *arbitrary* elements of $\mathbb{B}$ instead of just distinguishing the cases $a/b = 0$ and $a/b = 1$.

20pt

Problem 1.89: Given the following SML data types for Boolean formulae and truth values

```
datatype boolexp = zero | one
                 | plus of boolexp * boolexp
                 | times of boolexp * boolexp
                 | compl of boolexp
                 | var of int
datatype mybool = mytrue | myfalse
```

write a (cascading) evaluation function `eval : (int -> mybool) -> boolexp -> mybool` that takes an assignment φ and a Boolean formula e and returns $\mathcal{I}_\varphi(e)$ as a value.

20pt

Problem 1.90: Given the SML data types from ??, write a simplified version of the function using the built-in truth values in SML, i.e. an evaluation function `evalbib : (int -> bool) -> boolexp -> bool`. This function should not use any `if` constructs.

40pt

Problem 1.91 (Parsing boolean expressions)
Given the following SML data types for Boolean formulae

```
datatype boolexp = bez | beo (* 0 and 1 *)
                 | bep of boolexp * boolexp (* plus *)
                 | bet of boolexp * boolexp (* times *)
                 | bec of boolexp (* complement *)
                 | bev of int (* variables *)
```

write an SML function `beparse : string -> boolexp` that takes a string as input and transforms it into an `boolexp` representation of this formula, if it is in E_{bool} and raises an exception if not.

Note: As there is no ASCII representation for the complement operation we used in the definition in class, we use `-(x)` for the complement of `x` in the input syntax. So the relevant clause in the definition is now:

- $E_{\text{bool}}^{i+1} := \{a, -(a), (a + b), (a * b) \mid a, b \in E_{\text{bool}}^{i}\}$

Hint: For this you will need to write a couple of auxiliary functions, e.g. to convert lists of characters into integers and strings. A main function will have to look at all the characters in turn and decide what to do next.

20pt

Problem 1.92: Write a function `beprint : boolexp -> string` that converts `boolexp` formulae from ?? to E_{bool} strings. This should be the inverse function to the function `beparse` from Problem 1.91.

Test your implementation by round-tripping (check on some examples whether `beparse(beprint(x))=x` and `beprint(beparse(x))=x`). Exhibit at least three examples with at least 8 operators each, and show the results on them.

Problem 1.93: Is the expression $e := \overline{x123 * x72} + x123 * x4$ valid, satisfiable, unsatisfiable, falsifiable? Justify your answer. 3pt 5min

Problem 1.94 (Evaluating Expressions) 2pt 7min

Let $e := \overline{x_1 + x_2} + (\overline{\overline{x_2} * x_3} + x_3 * x_4)$ and $\varphi := [F/x_1], [F/x_2], [T/x_3], [F/x_4]$, compute the value $\mathcal{I}_\varphi(e)$, give a (partial) trace of the computation.

Problem 1.95 (Boolean Equivalence)

Prove or refute the following equivalence:

$$\overline{x_1 * x_1 + \overline{\overline{x_1} + x_2}} \equiv (\overline{x_1} + x_2) * ((\overline{x_1} + \overline{x_2}) * (\overline{x_1} + \overline{x_1}))$$

For each step write down which equivalence rule you used (by equivalence rules we mean commutativity, associativity, etc.).

1.6.2 Boolean Functions

Problem 1.96 (Induced Boolean Function) 10pt

Determine the Boolean function f_e induced by the Boolean expression $e := (x1 + x2) * \overline{x1 * x3}$. Moreover determine the CNF and DNF of f_e.

Problem 1.97 (CNF and DNF)

Write the CNF and DNF of the boolean function that corresponds to the truth table below.

x_1	x_2	x_3	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

1.6.3 Complexity Analysis for Boolean Expressions

Problem 1.98 (Landau sets)

Order the landau sets below by specifying which ones are subsets and which ones are equal
(e.g.: $O(a) \subset O(b) \subset O(c) \equiv O(d) \subset O(e)...$)

$$O(n^2);\ O((n)!);\ O(|\sin n|);\ O(n^n);\ O(1);\ O(2^n);\ O(2n^2 + 2^{72})$$

Problem 1.99 (Relations among polynomials) 5pt 6min

Prove or refute that $O(n^i) \subseteq O(n^j)$ for $0 \leq i < j, n$ $(i, j, n \in \mathbb{N})$.

Problem 1.100: Determine for the following functions f and g whether $f \in O(g)$, or $f \in \Omega(g)$, or $f \in \Theta(g)$, explain your answers. 3pt 10min

f	g	f	g
4572	84	$n^3 + 3 * n$	n^3
$\log(n^3)$	$\log(n)$	$(n^2) - 2^2$	n^3
16^n	2^n	n^n	2^{n+1}

Problem 1.101 (Upper and lower bounds)

For each of the functions below determine whether $f \in O(g)$, $f \in \Omega(g)$ or $f \in \Theta(g)$. Briefly explain your answers.

1. $f(n) = 235,\ g(n) = 12$
2. $f(n) = n,\ g(n) = 16n$
3. $f(n) = \log_{10}(n),\ g(n) = 7n + 2$
4. $f(n) = 7n^3 + 4n - 2,\ g(n) = 3n^4 + 1$
5. $f(n) = \frac{\log_2(n)}{n},\ g(n) = \frac{n}{\log_2(n)}$
6. $f(n) = 8^n,\ g(n) = 2^n$
7. $f(n) = n^{\log_n(5)},\ g(n) = 2^n$
8. $f(n) = n^n,\ g(n) = (\log_n(3))(n)!$
9. $f(n) = \binom{n}{2},\ g(n) = \binom{n}{4}$

Problem 1.102: What is the time complexity of the following SML function? Take one evaluation step to be a creation of a head in function `unwork` and disregard other operations.

```
fun gigatwist lst = let

        fun unwork nil = nil |
            unwork(hd::tl) = hd::unwork(tl)

        fun nextwork(nil, _) = nil |
            nextwork(hd::tl, fnc) = fnc(lst)@nextwork(tl, fnc)

        fun nthwork 1 = unwork |
            nthwork n = let
               fun work arg = nextwork(arg, nthwork(n-1))
            in
             work
            end
in
         nthwork(length lst) lst
end
```

Problem 1.103 (Proof of Membership in Landau Set) 3pt 10min

Prove by induction or refute: the function $f(n) := n^n$ is in $O((n)!^2)$; i.e. there is a constant c such that $n^n \leq (n)!^2$ for sufficiently large n.

Hint:

1.6.4 The Quine-McCluskey Algorithm

Problem 1.104 (Quine-McCluskey) 14pt

Execute the QMC algorithm for the following function:

x_1	x_2	x_3	f
F	F	F	T
F	F	T	T
F	T	F	F
F	T	T	T
T	F	F	T
T	F	T	F
T	T	F	T
T	T	T	T

Moreover you are required to find the solution with minimal cost where each operation (and, not, or) adds 1 to the cost. E.g. the cost of $(\overline{x_1} + x_3)\,(x_3)$ is 3. 35pt

Problem 1.105: Use the algorithm of Quine-McCluskey to determine the minimal polynomial of the following functions:

$x1$	$x2$	$x3$	$x4$	f_1
F	F	F	F	F
F	F	F	T	F
F	F	T	F	T
F	F	T	T	T
F	T	F	F	T
F	T	F	T	T
F	T	T	F	T
F	T	T	T	T
T	F	F	F	T
T	F	F	T	F
T	F	T	F	F
T	F	T	T	T
T	T	F	F	T
T	T	F	T	F
T	T	T	F	F
T	T	T	T	F

$x1$	$x2$	$x3$	$x4$	f_2
F	F	F	F	T
F	F	F	T	F
F	F	T	F	T
F	F	T	T	F
F	T	F	F	F
F	T	F	T	F
F	T	T	F	F
F	T	T	T	T
T	F	F	F	T
T	F	F	T	T
T	F	T	F	F
T	F	T	T	F
T	T	F	F	F
T	T	F	T	F
T	T	T	F	F
T	T	T	T	T

15pt

Problem 1.106 (Quine-McCluskey with Don't-Cares)

How can the Quine-McCluskey algorithm be modified to take advantage of don't-cares? Find out which steps of the algorithm are affected by this modification and explain how they change by showing the respective steps of applying the algorithm to the function $f(x1, x2, x3, x4)$ that yields T for $x1^0\, x2^1\, x3^0\, x4^0$, $x1^0\, x2^1\, x3^0\, x4^1$, $x1^0\, x2^1\, x3^1\, x4^0$, $x1^1\, x2^0\, x3^0\, x4^0$, $x1^1\, x2^0\, x3^0\, x4^1$, $x1^1\, x2^0\, x3^1\, x4^0$, $x1^1\, x2^1\, x3^0\, x4^1$, "don't care" for $x1^0\, x2^0\, x3^0\, x4^0$, $x1^0\, x2^1\, x3^1\, x4^1$, $x1^1\, x2^1\, x3^1\, x4^1$, and F for the other inputs.

14pt
12min

Problem 1.107 (CNF with Quine-McCluskey)

In class you have learned how to derive the optimal formula for a given function in DNF form using the Quine-McCluskey algorithm. It appears that the same algorithm could be applied to find the optimal formula in CNF form. Think of how this can be done and apply it on the function defined by the following table:

x_1	x_2	x_3	f
F	F	F	T
F	F	T	T
F	T	F	T
F	T	T	F
T	F	F	T
T	F	T	T
T	T	F	F
T	T	T	F

Hint:
The basic rule used in the QMC algorithm: $a\,x + a\,\overline{x} = a$ also applies for formulas in CNF: $(a + x)\,(a + \overline{x}) = (a)$

1.6.5 A simpler Method for finding Minimal Polynomials

10pt

Problem 1.108 (Karnaugh-Veitch Minimization)

Given the boolean function $f = B * \overline{D + C} + \overline{B} * \overline{(D + \overline{A}) * (A + D)}$:

1. Use a KV map to determine the minimal polynomial for the function.

2. Try to further reduce the cost of the resulting polynomial using boolean equivalences. The result does not need to be a polynomial.

3. Using boolean equivalences, transform the original expression into the the result from (2). Show all intermediate steps.

Problem 1.109 (Karnaugh-Veitch Diagrams) 10min

1. Use a KV map to determine all possible minimal polynomials for the function defined by the following truth table:

A	B	C	D	f
F	F	F	F	F
F	F	F	T	T
F	F	T	F	T
F	F	T	T	F
F	T	F	F	T
F	T	F	T	F
F	T	T	F	T
F	T	T	T	T
T	F	F	F	T
T	F	F	T	T
T	F	T	F	F
T	F	T	T	T
T	T	F	F	T
T	T	F	T	T
T	T	T	F	F
T	T	T	T	T

2. How would you use a KV map to find a minimal polynomial for a function with 5 variables? What does your map look like? Which borders in the map are virtually connected? (A simple but clear explanation suffices.)

Problem 1.110 (CNF with Karnaugh-Veitch Diagrams) 15pt 6min

KV maps can also be used to compute a minimal CNF for a Boolean function. Using the function $f(x1, x2, x3)$ that yields T for $x1^0\,x2^0\,x3^0$, $x1^0\,x2^1\,x3^0$, $x1^0\,x2^1\,x3^1$, $x1^1\,x2^0\,x3^0$, and F for the other inputs, develop an idea (and verify it for this example!) how to do this.

Hint: Start by grouping F-cells together.

Problem 1.111 (Karnaugh-Veitch Diagrams with Don't-Cares) 10pt

In some cases, there is an input $d \in \mathbf{dom}(f)$ to a boolean function $f\colon \mathbb{B}^n \to \mathbb{B}$ for which no output is specified — because the input is invalid or it would never occur. In a truth table for f, a function value $f(d)$ would be written as X instead of F or T, which means, "Don't care!"

Describe how don't-cares can be utilized when determining the minimal polynomial of a Boolean function using a KV map.

Note: Considering don't-cares is particularly beneficial when designing digital circuits. This will be done in GenCS 2. Just consider an electronic device with six states, which we can conveniently encode by using three boolean memory elements, which leads to $2^3 - 6 =$ two leftover "don't-care" states.

Problem 1.112 (Don't-Care Minimization) 10pt

1. Devise a concrete Boolean function $f\colon \mathbb{B}^4 \to \mathbb{B}$ that gives T for 6 of the 16 possible inputs, F for 7 inputs, and "don't care" for the remaining 3 possible inputs.

2. Apply the don't-care minimization algorithm from the previous exercise to it.

3. Then replace all don't-cares by T, do minimization without don't-cares, compare, and give a short comment.

1.7 Propositional Logic

1.7.1 Boolean Expressions and Propositional Logic

Problem 1.113 (The *Nor* Connective) 2pt 7min

All logical binary connectives can be expressed by the $\downarrow$ (*nor*) connective which is defined as $\mathbf{A} \downarrow \mathbf{B} := \neg(\mathbf{A} \vee \mathbf{B})$. Rewrite $\mathbf{P} \vee \neg\mathbf{P}$ (tertium non datur) into an expression containing only $\downarrow$ as a logical connective.

Hint: Recall that $\neg\mathbf{A} \Leftrightarrow \mathbf{A} \downarrow \mathbf{A}$.

1.7.2 Logical Systems and Calculi

Problem 1.114 (Calculus Properties)

Explain briefly what the following properties of calculi mean:

- correctness
- completeness

1.7.3 Proof Theory for the Hilbert Calculus

Problem 1.115: We have proven the correctness of the Hilbert calculus $\mathcal{H}^0$ in class. The problems of this quiz is about two incorrect calculi $\mathcal{C}^1$ and $\mathcal{C}^2$ which differ only slightly from $\mathcal{H}^0$. 5pt

What makes them incorrect?

Hint: The fact that $\mathcal{H}^0$ has two axioms, but each of $\mathcal{C}^1$ and $\mathcal{C}^2$ only have one is not the point. Remember the properties of axioms and inference rules which are preconditions for a correct calculus.

Why is this calculus $\mathcal{C}^1$ incorrect?

- $\mathcal{C}^1$ Axiom:$P \Rightarrow P \wedge Q$
- $\mathcal{C}^1$ Inference Rules: $\dfrac{\mathbf{A} \Rightarrow \mathbf{B} \quad \mathbf{A}}{\mathbf{B}}\,\mathrm{MP}$ $\qquad$ $\dfrac{\mathbf{A}}{[\mathbf{B}/P]\mathbf{A}}\,\mathrm{Subst}$

Why is this calculus $\mathcal{C}^2$ incorrect?

- $\mathcal{C}^2$ Axiom: $P \Rightarrow (Q \Rightarrow P)$
- $\mathcal{C}^2$ Inference Rules: $\dfrac{\mathbf{A} \vee \mathbf{B} \quad \mathbf{A}}{\mathbf{A} \wedge \mathbf{B}}\,R2$ $\qquad$ $\dfrac{\mathbf{A}}{[\mathbf{B}/P]\mathbf{A}}\,\mathrm{Subst}$

Problem 1.116 (Almost a Proof)

Please consider the following sequence of formulae: it pretends to be a proof of the formula $\mathbf{A} \Rightarrow \mathbf{A}$ in $\mathcal{H}^0$. For each line annotate how it is derived by the inference rules from proceeding lines or axioms. If a line is not derivable in such a manner then mark it as underivable and explain what went wrong.

Use the aggregate notation we used in the slides for derivations with multiple steps (e.g. an axiom with multiple applications of the Subst rule)

1. $\mathbf{A} \Rightarrow (\mathbf{B} \Rightarrow \mathbf{A})$
2. $\mathbf{B} \Rightarrow \mathbf{A}$
3. $\mathbf{B} \Rightarrow (\mathbf{A} \Rightarrow \mathbf{B})$

4. $\mathbf{A} \Rightarrow \mathbf{B}$

5. $(\mathbf{B} \Rightarrow \mathbf{A}) \Rightarrow (\mathbf{A} \Rightarrow (\mathbf{B} \Rightarrow \mathbf{A}))$

6. $(\mathbf{A} \Rightarrow (\mathbf{B} \Rightarrow \mathbf{A})) \Rightarrow ((\mathbf{A} \Rightarrow \mathbf{B}) \Rightarrow (\mathbf{A} \Rightarrow \mathbf{A}))$

7. $(\mathbf{A} \Rightarrow \mathbf{B}) \Rightarrow (\mathbf{A} \Rightarrow \mathbf{A})$

8. $\mathbf{A} \Rightarrow \mathbf{A}$

Problem 1.117: We have proven the correctness of the Hilbert calculus $\mathcal{H}^0$ in class. The problems of this quiz is about two incorrect calculi $\mathcal{C}^1$ and $\mathcal{C}^2$ which differ only slightly from $\mathcal{H}^0$. What makes them incorrect?

Hint: The fact that $\mathcal{H}^0$ has two axioms, but each of $\mathcal{C}^1$ and $\mathcal{C}^2$ only have one is not the point. Remember the properties of axioms and inference rules which are preconditions for a correct calculus.

Why is this calculus $\mathcal{C}^1$ incorrect?

- $\mathcal{C}^1$ Axiom: $P \Rightarrow (Q \Rightarrow R)$

- $\mathcal{C}^1$ Inference Rules: $\dfrac{\mathbf{A} \Rightarrow \mathbf{B} \quad \mathbf{A}}{\mathbf{B}}\,\mathrm{MP} \qquad\qquad \dfrac{\mathbf{A}}{[\mathbf{B}/P]\mathbf{A}}\,\mathrm{Subst}$

Problem 1.118 (Alternative Calculus)
Consider a calculus given by the axioms $\mathbf{A} \vee \neg\mathbf{A}$ and $\mathbf{A} \wedge \mathbf{B} \Rightarrow \mathbf{B} \wedge \mathbf{A}$ and the following rules:

$$\frac{\mathbf{A} \Rightarrow \mathbf{B}}{\neg\mathbf{B} \Rightarrow \neg\mathbf{A}}\,Transp \qquad\qquad \frac{\mathbf{A}}{[\mathbf{B}/P]\mathbf{A}}\,\mathrm{Subst}$$

Prove that the calculus is sound.

Problem 1.119 (A calculus for propositional logic) 10pt 10min
Let us assume a calculus for propositional logic that consists of the single axiom $\mathbf{A} \Rightarrow \mathbf{A}$ and the inference rule:

$$\frac{\mathbf{A} \Rightarrow (\mathbf{B} \Rightarrow \mathbf{C})}{\mathbf{A} \wedge \mathbf{B} \Rightarrow \mathbf{C}} \qquad\qquad \frac{\mathbf{A}}{[\mathbf{B}/P]\mathbf{A}}\,\mathrm{Subst}$$

1. Show that this calculus is sound (i. e. correct).

2. Prove the formula $((P \Rightarrow Q) \wedge P) \Rightarrow Q$ using this calculus.

Problem 1.120 (Hilbert Calculus)
Prove the following theorem using $\mathcal{H}^0$: $((\mathbf{A} \Rightarrow \mathbf{C}) \Rightarrow \mathbf{A}) \Rightarrow ((\mathbf{A} \Rightarrow \mathbf{C}) \Rightarrow ((\mathbf{B} \Rightarrow \mathbf{B}) \Rightarrow \mathbf{A}))$

Problem 1.121 (A Hilbert Calculus) 20pt
Consider the Hilbert-style calculus given by the following axioms:

1. $(\mathbf{F} \vee \mathbf{F}) \Rightarrow \mathbf{F}$ (idempotence of disjunction)

2. $\mathbf{F} \Rightarrow (\mathbf{F} \vee \mathbf{G})$ (weakening)

3. $(\mathbf{G} \vee \mathbf{F}) \Rightarrow (\mathbf{F} \vee \mathbf{G})$ (commutativity)

4. $(\mathbf{G} \Rightarrow \mathbf{H}) \Rightarrow ((\mathbf{F} \vee \mathbf{G}) \Rightarrow (\mathbf{F} \vee \mathbf{H}))$

and the identities

1. $\mathbf{A} \Rightarrow \mathbf{B} = \neg\mathbf{A} \vee \mathbf{B}$
2. $\mathbf{F} \wedge \mathbf{G} = \neg(\neg\mathbf{F} \vee \neg\mathbf{G})$

You can use the MP and substitution as inference rules:

$$\frac{\mathbf{A} \Rightarrow \mathbf{B} \quad \mathbf{A}}{\mathbf{B}}\,\mathrm{MP} \qquad \frac{\mathbf{A}}{[\mathbf{B}/X](\mathbf{A})}\,\mathrm{Subst}$$

Prove the formula $\mathbf{P} \wedge \mathbf{Q} \vee (\mathbf{P} \vee (\neg\mathbf{P} \vee \neg\mathbf{Q}))$

1.7.4 The Calculus of Natural Deduction

No problems supplied yet.

1.8 Machine-Oriented Calculi

1.8.1 Calculi for Automated Theorem Proving: Analytical Tableaux

Problem 1.122: Prove the Hilbert-Calculus axioms $P \Rightarrow (Q \Rightarrow P)$, and $(P \Rightarrow (Q \Rightarrow R)) \Rightarrow ((P \Rightarrow Q) \Rightarrow (P \Rightarrow R))$

Problem 1.123: Prove the associative law for disjunction $(P \vee Q) \vee R \Leftrightarrow P \vee (Q \vee R)$[2] with the tableau method.

0pt

Problem 1.124 (Tableau Calculus) 10min

1. Explain the difference between tableau proof of validity and model generation.
2. Derive a tableau inference rule for $A \Leftrightarrow B^{\mathsf{T}}$. Show the derivation.
3. Generate all models of the following expression: $\neg Q \wedge P \Leftrightarrow Q \wedge \neg P$

11pt

Problem 1.125 (Refutation and model generation in Tableau Calculus)

1. Prove the following proposition:

$$\models \neg A \wedge \neg B \Rightarrow \neg(A \vee B)$$

2. Find all models for the following proposition:

$$\models (A \Rightarrow B) \wedge (B \Rightarrow A \wedge B)$$

Hint: You may use derived rules for implication and disjunction.

14pt

Problem 1.126 (Tableau Calculus)

Prove or refute that the following proposition is valid using a tableaux:

$$(P \Rightarrow Q) \vee R \Leftrightarrow \neg R \wedge Q \Rightarrow S$$

4pt

Problem 1.127 (A *Nor* Tableau Calculus) 13min

Develop a variant of the tableau calculus presented in class for propositional formulae expressed with $\downarrow$ (i.e. "not or") as the only logical connective.

Complete the following scheme of inference rules for such a tableau calculus and proof its correctness

$$\frac{\mathbf{A} \downarrow \mathbf{B}^{\mathsf{T}}}{?} \qquad \frac{\mathbf{A} \downarrow \mathbf{B}^{\mathsf{F}}}{?} \qquad \frac{\begin{array}{c}\mathbf{A}^{\alpha} \\ \mathbf{A}^{\beta}\end{array} \quad \alpha \neq \beta}{\perp}$$

Prove the formula $(P \downarrow (P \downarrow P)) \downarrow (P \downarrow (P \downarrow P))$ in your new tableau calculus.

35pt

Problem 1.128 (Tableau Construction)

Write an SML function that computes a complete tableau for a labeled formula. Use the data type `prop` for formulae and the datatype `tableau` for tableaux.

```
datatype prop = tru | fals (* true and false *)
              | por of prop * prop (* disjunction *)
              | pand of prop * prop (* conjunction *)
              | pimpl of prop * prop (* implication *)
              | piff of prop * prop (* biconditional *)
              | pnot of prop (* negation *)
              | var of int (* variables *)
```

[2]Proving this in the Hilbert calculus from ?? takes about 300 steps.

```
datatype label = prove | refute
datatype tableau = ext of prop * label * tableau (* extension by a formula *)
                 | cases of tableau * tableau (* two branches *)
                 | complete (* branch completehalt *)
```

Hint: Write a recursive function `ctab` that takes a list of (unresolved) proposition/label pairs as an input, goes through them, extending the tableau as needed.

Problem 1.129 (Automated Theorem Prover) 30pt
Building on the tableau procedure from Problem 1.128 build an automated theorem prover for propositional logic. Concretely build an SML function `prove` that given a formula F outputs `valid`, if F is valid, and returns a counterexample otherwise (i.e. an interpretation of the variables that satisfy F^{T}).

Problem 1.130 (Testing the ATP) 30pt
Use the random formula generators from ?? to test your tableau implementation. Run experiments on large sets (e.g. 100) of random formulae with differing depths and plot the runtimes, percentages of valid formulae, over depths, and weights, and variable numbers. Interpret the results briefly.

Hint: You can use any plotting software you are familiar with, e.g. Excel or gnuplot. If you are not familiar with any, use pen and paper. Do not waste time on the plotting aspect.

G

1.8.2 Resolution for Propositional Logic

Problem 1.131: Compute the Clause normal form of $(P \Leftrightarrow Q) \Leftrightarrow (R \Leftrightarrow P)$ with and without using the derived rules. 10pt

Problem 1.132: Prove in the resolution calculus using derived rules:

$$\models A \wedge (B \vee C) \Rightarrow (A \wedge B \vee A \wedge C)$$

Problem 1.133 (Basics of Resolution) 4pt 8min
What are the principal steps when you try to prove the validity of a propositional formula by means of resolution calculus? In case you succeed deriving the empty clause, why does this mean you have found a proof for the validity of the initial formula?

Problem 1.134 (Resolution Calculus with Nand Connective) 5pt
Develop a variant $PropCNFCalcNAND$ of the CNF transformation calculus presented in class that transforms propositional formulae expressed with $NAND$ (denoted by $\uparrow$) as the only logical connective. To do so just complete the scheme of inference rules given here:

$$\frac{\mathbf{C} \vee \mathbf{A} \uparrow \mathbf{B}^{\mathsf{T}}}{?} \qquad \frac{\mathbf{C} \vee \mathbf{A} \uparrow \mathbf{B}^{\mathsf{F}}}{?}$$

With this variant $\mathcal{CNF}^{\uparrow}$ together with the usual inference rule from resolution calculus conduct a resolution proof to verify the formula $(A \uparrow A) \uparrow ((A \uparrow B) \uparrow (A \uparrow B))$

Problem 1.135: Use the resolution method to prove the formulae from ??: 25pt

1. $(\neg P \Rightarrow Q) \Rightarrow ((P \Rightarrow Q) \Rightarrow Q)$
2. $(P \Rightarrow Q) \wedge (Q \Rightarrow R) \Rightarrow \neg(\neg R \wedge P)$

You may use any derived correctly derived inference rules such as for instance:

$$\frac{\mathbf{A} \Rightarrow \mathbf{B}^{\mathsf{F}}}{\begin{array}{c}\mathbf{A}^{\mathsf{T}} \\ \mathbf{B}^{\mathsf{F}}\end{array}}$$

However, if you use more complex inference rules (i.e. more than one connective involved) then you have to prove your derived inference rule.

Problem 1.136: Consider the following two formulae where the first one is in conjunctive normal form and the second in disjunctive normal form 25pt

1. $(P \vee \neg P) \wedge (Q \vee \neg Q)$
2. $P \wedge Q \vee (\neg P \vee \neg Q)$

Try to find the shortest proofs of both formulae using the resolution method as well as the tableau method. Describe your observations concerning the proof length in dependency on the normal form and proof method.

2 How to build Computers and the Internet (in principle)

2.1 Circuits

2.1.1 Graphs and Trees

Conjecture 5 *Let G be a graph with a cycle and $n \in \mathbb{N}$, then there is a path p in G with $length(p) > n$.*

20pt

Problem 2.1 (Infinite Paths)
Prove or refute ?? using the formal definitions (no, it is not sufficient to just draw a picture).

25pt

Problem 2.2 (Node Connectivity Relation is an Equivalence Relation)
Let $G = \langle V, E \rangle$ be an undirected graph and the relation C be defined as

$$C := \{\langle u, v \rangle \mid \text{there is a path from } u \text{ to } v\}$$

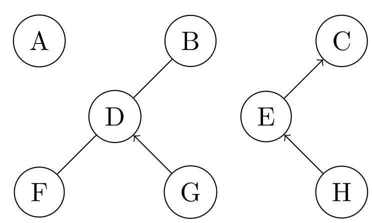
Prove or refute that C is an equivalence relation.

Hint: Recall the properties of an equivalence relation!

Problem 2.3 (Directed Graph)
We call a graph connected, iff for any two nodes n_1 and n_2 there is a path starting at n_1 and ending at n_2.

Complete the partially directed graph below by adding directed edges or directing undirected edges such that it becomes a connected, (fully) directed graph where each $indeg(n) = outdeg(n)$ for all nodes n.

How many initial, terminal nodes and how many paths does your graph have?



4pt

Problem 2.4: Draw examples of

1. a directed graph with 4 nodes and 6 edges
2. a undirected graph with 7 nodes and 8 edges.

Present a mathematical representation of these graphs.

Problem 2.5 (Planar Graphs)
A graph G is called planar if G can be drawn in the plane in such a manner that edges do not cross elsewhere than vertices. The geometric realization of a planar graph gives rise to regions in the plane called faces; if G is a finite planar graph, there will be one unbounded (i.e. infinite) face, and all other faces (if there are any) will be bounded. Given a planar realization of the graph G, let $v = \#(V)$, $e = \#(E)$, and let f be the number of faces (including the unbounded face) of G's realization.

Prove or refute the Euler formula, i.e. that $v - e + f = 2$, must hold for a connected planar graph.

Problem 2.6 (Parse trees and isomorphism)
Let P_e be the parse-tree of $e := \overline{x_1} + (x_2 + x_3) * x_4$

1. Draw the graphic representation of P_e.
2. Write the mathematical representation of a graph G that is different but equivalent to P_e.

3pt

Problem 2.7 (Size and Depth of a Binary Tree)
Given the following data type for binary trees, define functions `size` and `depth` that compute the depth and the size of a given tree.

```
datatype btree = leaf | parent of btree * btree
```

Write a function `fbbtree` that given a natural number n returns a fully balanced binary tree of depth n

Problem 2.8 (Graph basics)
For each of the five directed graphs below do the following:

- State whether the graph is also a tree and explain why.
- Determine the depth of the graph.
- Write out in math notation a path from A to E if one exists and determine the path's length.

1. $G_1 := \langle\{A,B,C,D,E\},\{\langle A,B\rangle,\langle A,C\rangle,\langle A,D\rangle,\langle D,E\rangle\}\rangle$
2. $G_2 := \langle\{A,B,C,D,E\},\{\langle A,B\rangle,\langle B,C\rangle,\langle C,A\rangle,\langle C,D\rangle,\langle C,E\rangle\}\rangle$
3. $G_3 := \langle\{A,B,C,D,E\},\{\langle A,B\rangle,\langle B,C\rangle,\langle B,D\rangle,\langle C,E\rangle\}\rangle$
4. $G_4 := \langle\{A,B,C,D,E\},\{\langle A,B\rangle,\langle A,C\rangle,\langle B,D\rangle,\langle D,C\rangle,\langle C,B\rangle,\langle A,D\rangle\}\rangle$
5. $G_5 := \langle\{A,B,C,D,E\},\{\langle D,A\rangle,\langle D,B\rangle,\langle D,E\rangle,\langle D,C\rangle\}\rangle$

Conjecture 6 *1. Let $G = \langle V,E\rangle$ be a directed graph. Then,*

$$\sum_{i=1}^{\#(V)} indeg(v_i) = \sum_{i=1}^{\#(V)} outdeg(v_i) = \#(E)$$

2. If G is undirected, we have

$$\sum_{i=1}^{\#(V)} deg(v_i) = 2\cdot\#(E)$$

25pt

Problem 2.9 (Degrees in an Undirected Graph)
Prove or refute the conjecture above

Note: For undirected graphs, we introduce the notation deg with $\deg(v) = \mathrm{indeg}(v) = \mathrm{outdeg}(v)$ for each node.

Hint: Use induction over the number of edges. Derive the second assertion from the first one.

Problem 2.10 (Graph representation in memory)
How would you represent a graph in memory if you write a program which processes it in some way? Give 2-3 variants and explain the advantages and disadvantages of each method.

4pt

Problem 2.11: How many edges can a directed graph of size n (i.e. with n vertices) have maximally. How many can it have if it is acyclic? Justify your answers (prove them).

4pt

Problem 2.12 (Undirected tree properties)
We've defined the notion of *path* for the directed graphs.

- Define the notion of *path* and *cycle* for the undirected graphs.

We call an undirected graph connected, iff for any two nodes $n_1 \neq n_2$ there is a path starting at n_1 and ending at n_2.

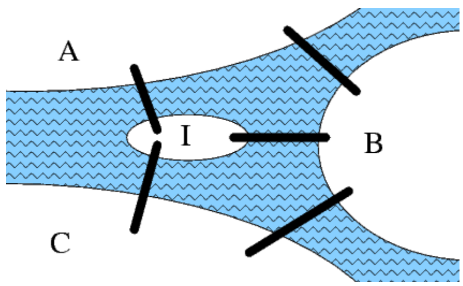
An **undirected tree** is an undirected acyclic connected graph.

Let $G = \langle V,E\rangle$. Prove or refute that the following statements are equivalent:

1. G is an undirected tree
2. For any two nodes $n_1 \neq n_2$ there is a *single* path starting at n_1 and ending at n_2
3. G is a connected graph, but it becomes disconnected after deleting any edge
4. G is connected and $\#(E) = \#(V) - 1$
5. G is acyclic and $\#(E) = \#(V) - 1$
6. G is acyclic, but adding one edge to E introduces a cycle

Problem 2.13 ((Modified) Königsberg Bridge Problem) 3pt 10min

Consider a river fork with three banks (A,B,C) and one island (I) connected with bridges as shown in the figure.



Is it possible to walk accross each of the bridges exactly once in an uninterrupted tour and return to the starting point?

In order to prove your answer first translate the question into a graph problem where the banks and the island are modeled as nodes and the bridges as undirected edges.

Hint: Consider the degree of each node (i.e.the number for edges connected to it). Relate the degrees of the nodes to the constraint of an uninterrupted tour.

Problem 2.14 (Parse Tree) 35pt

Given the data type prop for formulae

```
datatype prop = tru | fals (* true and false *)
             | por of prop * prop (* disjunction *)
             | pand of prop * prop (* conjunction *)
             | pimpl of prop * prop (* implication *)
             | piff of prop * prop (* biconditional *)
             | pnot of prop (* negation *)
             | var of int (* variables *)
```

Write an SML function that computes the parse tree for a formula. The output format should be

- a list of integers for the set of vertices,
- a list of pairs of integers for the set of edges,
- and for the labeling function a list of pairs where the first component is an integer and the second a string (the label).

2.1.2 Introduction to Combinatorial Circuits

Problem 2.15 (DNF Circuit with Quine McClusky) 25pt

Use the technique shown in class to design a combinational circuit for the following Boolean function:

X_1	X_2	X_3	$f_1(X)$	$f_2(X)$	$f_3(X)$
0	0	0	1	0	0
0	0	1	1	1	1
0	1	0	1	0	0
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	0	0	1
1	1	0	1	1	1
1	1	1	1	1	1

Hint: Use Quine-McCluskey to compute minimal polynomials for the three component functions, look for shared monomials, and build the DNF circuit.

Problem 2.16 (DNF Circuit with Quine McCluskey) 25pt

Use the technique shown in class to design a combinational circuit for the following Boolean function:

X_1	X_2	X_3	$f_1(X)$	$f_2(X)$	$f_3(X)$
0	0	0	0	1	1
0	0	1	0	0	0
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	1	1	0
1	1	0	0	0	0
1	1	1	0	0	0

Hint: Use Quine-McCluskey to compute minimal polynomials for the three component functions, look for shared monomials, and build the DNF circuit.

Problem 2.17 (Combinational Circuit) 12pt 10min

Consider the following Boolean function

$$f\colon \{0,1\}^3 \to \{0,1\}^2; \langle i_1, i_2, i_3\rangle \mapsto \langle \overline{i_1} * i_2 + i_2 * \overline{i_3}, \overline{i_1 + i_2} * i_3\rangle$$

Draw the corresponding combinational circuit and write down its labeled graph $G = \langle V, E, f_g\rangle$ in explicit math notation.

Problem 2.18 (Combinational Circuit for Shift) 5pt 10min

Design an explicit 4-bit shifter (combinational circuit) (using only NOT, AND and OR gates) that corresponds to $f_{\text{shift}}\colon \mathbb{B}^4 \times \mathbb{B} \times \mathbb{B} \to \mathbb{B}^4$ with

$$f_{\text{shift}}(\langle a_3, a_2, a_1, a_0\rangle, s_1, s_2) \begin{cases} \langle a_3, a_2, a_1, a_0\rangle & \text{if } s_1 = 0, s_2 = 0 \\ \langle a_2, a_1, a_0, 0\rangle & \text{if } s_1 = 1, s_2 = 0 \\ \langle 0, a_3, a_2, a_1\rangle & \text{if } s_1 = 0, s_2 = 1 \\ \langle a_0, a_3, a_2, a_1\rangle & \text{if } s_1 = 1, s_2 = 1 \end{cases}$$

Hint: Think of a variant of multiplexer.

Problem 2.19 (Is XOR universal?)

Imagine a logical gate XOR that computes the logical exclusive disjunction. Prove or refute whether the set $S = \{\text{XOR}\}$ is *universal*, considering the following two cases:

1. combinational circuits without constants

2. combinational circuits with constants

If the set turns out to be *not* universal in either of the cases, add *one* appropriate non-universal gate $G \in \{\mathrm{AND}, \mathrm{OR}, \mathrm{NOT}\}$ to S, and prove that the set $S' = \{\mathrm{XOR}, G\}$ is universal.

Note: A set of Boolean function is called *universal* (also called "functionally complete"), if *any* Boolean function can be expressed in terms of the functions from that set. $\{\mathrm{NAND}\}$ is an example from the lecture.

Problem 2.20 (Alarm System)

You have to devise an alarm system that signals if the image recorded by a camera changes. The camera is preprogrammed with a static image, divided into 8 regions. Whenever an observed region is different from the preprogrammed one, the corresponding input bit $\langle r_0, ..., r_7\rangle$ is set to 1. The image is sampled at discrete time periods. The value of an input (clk) changes between 0 and 1 on every time interval.

Design a circuit with one output which is set to 1 if two or more regions (the inputs $\langle r_0, ..., r_7\rangle$) are different from the preprogrammed image for two consecutive intervals. We do not care if different sets of regions are marked as different between the consecutive intervals. We also don't care what happens once the output is set to one.

You may use all elementary gates and all circuit blocks studied in class.

Hint:

- First make a circuit that determines how many of the regions are different.
- Make a circuit that outputs 1 if two or more regions are different in 2 consecutive intervals.

2.1.3 Realizing Complex Gates Efficiently

999pt

Balanced Binary Trees **Problem 2.21 (Operations on Binary Trees)**

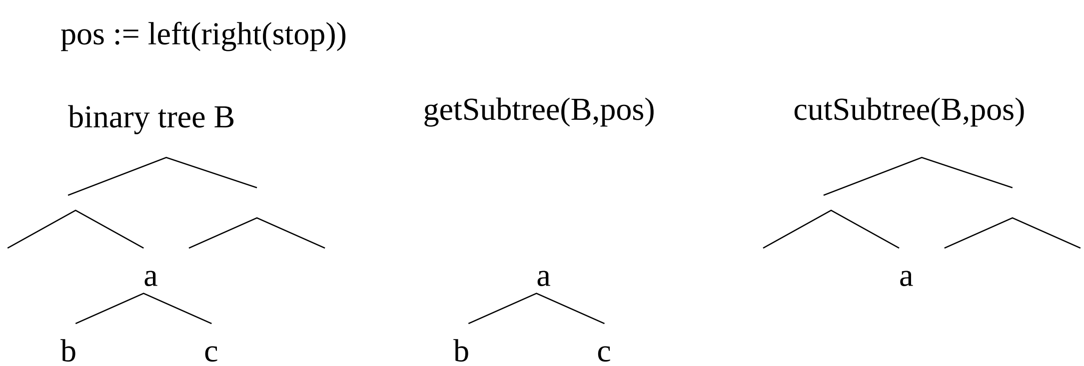
Given the SML datatype `btree` for binary trees and `position` for a position pointer into a binary tree:

```
datatype btree = leaf | parent of btree * btree;
datatype position = stop | right of position | left of position;
```

The interpretation of a position `right(left(stop))` is like a reversed path: start from root follow the right branch then the left and then stop.

Write two SML functions:

- `getSubtree` that takes a binary tree and a position and returns the subtree of the that binary at the corresponding position.
- `cutSubtree` that takes a binary tree and a position and returns the binary tree where the subtree at the corresponding position is cut off; i.e replaced by a leaf.



In both cases an exception should be raised if the position exceeds the observed binary tree. 4pt

Problem 2.22 (Number of Paths in Balanced Binary Tree)

Let $p(n)$ be the number of different paths in a fully balanced binary tree of depth n. Find a recursive equation for $p(n)$.

Hint: Do not forget the base case(s) for small n.

Problem 2.23 (Length of the inner path in balanced trees) 4pt

Prove by induction or refute that in a balanced binary tree the length of the inner path is not more than $(n+1)\lfloor\log_2(n)\rfloor - 2 \cdot 2^{\lfloor\log_2(n)\rfloor} + 2$. Here n is the number of nodes in the graph.

Note: Length of the inner path is the sum of all lengths of paths from the root to the nodes.

Problem 2.24 (Depth of a Fully Balanced Binary Tree) 10pt

Prove or refute that in a fully balanced binary tree with $n \geq 1$ nodes, the depth is $\log_2(n)$.

Realizing n-ary Gates No problems supplied yet.

2.2 Arithmetic Circuits

2.2.1 Basic Arithmetics with Combinational Circuits

6pt **Positional Number Systems** 6min

Problem 2.25 (Number System Conversion)
Convert the following 12-bit twos complement numbers into hexadecimal and decimal numbers.

1. 1000 0101 0100
2. 0010 1000 1010
3. 1101 0101 1001

25pt

Problem 2.26 (Mapping between Positional Number Systems)
Show that the mapping $\psi\colon D^+ \to \{/\}^*$ from the definition of a positional number system is indeed a bijection.

20pt

Problem 2.27 (Binary Number Conversion)
Write an SML function `binary` that converts decimal numbers into binary strings and an inverse `decimal` that converts binary strings into decimal numbers. Use the positive integers (of built-in type `int`) as a representation for decimal numbers. `binary` should raise an exception, if applied to a negative integer.

Problem 2.28 (Playing with bases)
Convert 2748 from decimal to hexadecimal, binary and octal representation.

Problem 2.29 (Converting to decimal in SML)
Write an SML function

```
to_int = fn : string -> int
```

that takes a string in binary, octal or hexadecimal notation and converts it to a decimal integer. If the string represents a binary number, it begins with 'b' (e.g. "b1011"), if it is an octal number - with '0' (e.g. "075") and if it is a hexadecimal number it begins with '0x' (e.g. "0x3A").

If the input does not represent an integer in one of these three forms raise the `InvalidInput` exception.

For example we have

```
to_int("b101010") -> 42
```

Adders Problem 2.30 (Cost and depth of adders)
What is the cost and depth of an n-bit CCA? What about the n-bit CSA (for cost, big-O is enough)? Now what if we construct a new adder, that computes the two cases for the first half of the input just like CSAs do (and of course uses a multiplexer), but only does this once, and the $\frac{n}{2}$-bit adders are not also CSAs, but CCAs (so only one multiplexer is used overall) - what would the cost and depth of this adder be?

35pt

Problem 2.31 (Carry Chain and Conditional Sum Adder)
Draw an explicit combinational circuit of a 4-bit Carry Chain Adder and a 4-bit Conditional Sum Adder. Do not use abbreviations, but only NOT, AND, OR, XOR gates. Demonstrate the addition of the two binary numbers $\langle 1,0,1,1\rangle$ and $\langle 0,0,1,1\rangle$ on both adders; i.e. annotate the output of each logic gate of your adders with the result bit for the given two binary numbers as input of the whole adder.

4pt 10min

Problem 2.32 (Carry Chain Adder and Subtractor for TCN)

- Draw a 2-bit carry chain adder only using (1-bit) full adders.
- Draw a subtractor for two's complement numbers using (1-bit) full-adders and Boolean gates of your choice.

Hint: Remember: An n-bit subtractor $f^n_{\mathrm{SUB}}(a,b,b')$ can be implemented as n-bit full-adder ($\mathrm{FA}^n(a,\overline{b},b')$)

Problem 2.33 (Half Adder)
Design an explicit combinational circuit for the half-adder using only NOR gates. What is its cost and depth? Looking at the first, straightforward solution, can cost and depth be improved?

Hint: First express the XOR gate by AND, OR and NOT gates then express each of these gates by NOR gates. Then think of further improvements.

Problem 2.34 (2-Stage Adder)
Design a circuit that computes the sum of two 6-bit numbers. In your solution you can use only a single 3-bit Adder, you are not allowed to implement an additional adder using elementary gates. You have to perform the computation in two steps. Therefore an additional control input is available. At first it will be 0. Then it will be set to 1 (you do not have to implement this yourself). After that the output of your circuit should represent the sum of the two numbers including a carry bit. You may use all circuit gates and block from the lecture notes.

Hint: Think about using the D Flip-Flop with an enable input to store intermediate data.

2.2.2 Arithmetics for Two's Complement Numbers

Problem 2.35 (Binary Number Systems)

- Write down the definition of $\langle\!\langle\cdot\rangle\!\rangle$, $(\langle\!\langle\cdot\rangle\!\rangle^{-})$, and $\langle\!\langle n\rangle\!\rangle^{2s}_{\cdot}$.
- Given the binary number $a = 10110$ compute $\langle\!\langle a\rangle\!\rangle$, $(\langle\!\langle a\rangle\!\rangle^{-})$, and $\langle\!\langle a\rangle\!\rangle^{2s}_{n}$.

Problem 2.36 (Sign-and-Magnitude Adder)
Recall the naïve *sign and magnitude* representation for n-bit integers: If the sign bit is 0, the number is positive, else negative. The other $n-1$ bits represent the absolute value of the number.

1. Describe how to add two equally-signed n-bit numbers (simple).
2. Describe how to add two n-bit numbers numbers with different sign bits (a bit more tricky).
3. Draw a combinational circuit of a 4-bit sign and magnitude adder (one sign bit, three data bits). You may use the 1-bit full adder/subtractor (with one input that selects whether to add or to subtract) known from the lecture, an n-bit multiplexer that selects one of two n-bit numbers, as well as an n-bit comparator that computes the function $f\colon \{0,1\}^2 \to \{0,1\}$ defined as follows:
$$f(a,b) := \begin{cases} 1 & \text{if } a \leq b \\ ,0 & \text{else} \end{cases}$$
Be sure to explain the layout of your circuit.
4. How can an over-/underflow be detected at the outputs? In which cases can an over-/underflow occur?

Problem 2.37: Given following integer numbers in base ten. Convert them to 32-bit Two's Complement numbers. 12pt

1. 3643
2. 5731923
3. -128
4. -24689

Problem 2.38: Given the following integer numbers as 16-bit Two's Complement numbers. 9pt

1. 1010 0001 0100 0000

2. 0010 1110 1110 1110

3. 1101 0011 1111 0010

Convert them into decimal numbers. 4pt

Problem 2.39 (The Structure Theorem for TCN) 7min

Write down the structure theorem for two's complement numbers (TCN) and make use of it to convert

- the integer -53 into a 8-bit TCN.
- the 8-bit TCN 10110101 into an integer.

Furthermore convert

- the integer -53 into a 10-bit TCN.
- the 10-bit TCN 1110110101 into an integer.

The 10-bit version of the conversion task shouldn't be any effort after solving the 8-bit version. You just have to remember the appropriate lemma to transfer an n-bit TCN to an $n+1$-bit TCN. How is the lemma called and what does it state? 15pt

Problem 2.40 (2s Complement Conversion)

Write an SML function `tcn` that takes an integer i and a natural number n as arguments and converts i into an n-bit two's complement number if it is in range and raises an exception otherwise.

Write an SML function that converts a 2s complement number into a decimal integer. 8pt

Problem 2.41 (Shift and Duplication on PNS) 10min

Consider for this problem the signed bit number system and the two's complement number system. Given a binary string $b = a_n \ldots a_0$. We define

1. the duplication function *dupl* that duplicates the leading bit; i.e. it maps the $n+1$-bit number $a_n \ldots a_0$ to the $n+2$-bit number $a_n a_n \ldots a_0$ and
2. the shift function $shift$ that maps the $n+1$-bit number $a_n \ldots a_0$ to the $n+2$-bit number $a_n \ldots a_0 0$

Prove or refute the following two statements

- The $shift$ function has the same effect in both number systems; i.e. for any integer z:

$$(\langle\!\langle shift(B(z)) \rangle\!\rangle^{-}) = \langle\!\langle shift(B_n^{2s}(z)) \rangle\!\rangle_{n+1}^{2s}$$

- The *dupl* function has the same effect in both number systems; i.e. for any integer z:

$$(\langle\!\langle dupl(B(z)) \rangle\!\rangle^{-}) = \langle\!\langle dupl(B_n^{2s}(z)) \rangle\!\rangle_{n+1}^{2s}$$

15pt

Problem 2.42: Compute the intermediate carry ($\mathrm{ic}_k(9235, 26234, 1)$) for $k = 3$ and $k = 5$.

Hint: You have to convert the first two arguments to binary numbers of the same range beforehand.

2.2.3 Algorithmic/Logic Units

3pt

Problem 2.43 (TCN Substraction) 6min

Let $A = 576$ and $B = 9$.

1. convert the numbers into an n-bit TCN system. What is the minimal n in order to encode A as well as B?

2. perform a binary subtraction $A - B$ and check the result by converting back to the decimal system.

Problem 2.44 (Carry Chain Adder and Subtractor for TCN) 2pt 6min

- Draw a 2-bit carry chain adder only using (1-bit) full adders as primitives.
- Draw a 2-bit subtractor for two's complement numbers using (1-bit) full-adders and Boolean gates of your choice.

Hint: Remember: An n-bit subtractor $f^n_{\mathrm{SUB}}(a, b, b')$ can be implemented as n-bit full-adder $(\mathrm{FA}^n(a, \overline{b}, \overline{b'}))$

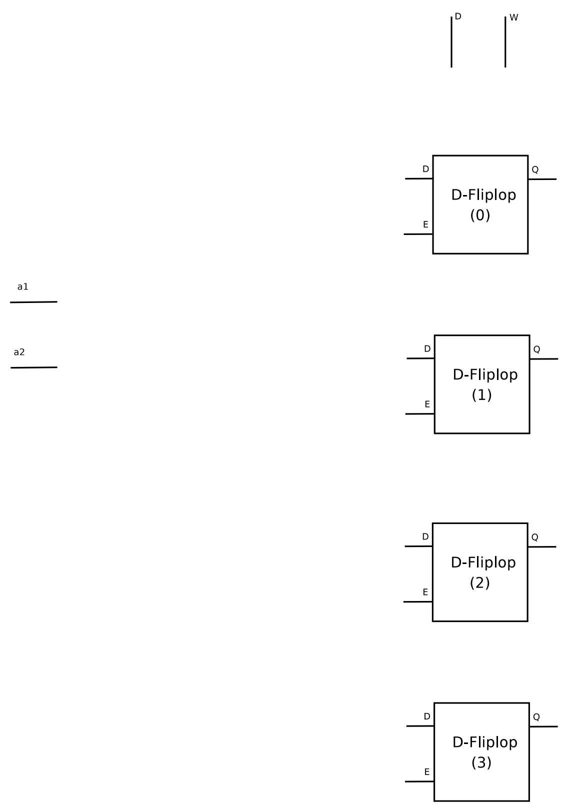
2.3 Sequential Logic Circuits and Memory Elements

Problem 2.45 (2bit Address Decoder)
Design a 2 bit address decoder using only NOR gates.

Problem 2.46 (Reading from and writing to memory)

Suppose you have a 2-bit addressed memory of 4 bits managed by 4 D-Flipflops aligned as shown in the figure. The input of the circuit consists of a total of 4 bits. 2 of the bits (a_0 and a_1) provide a 2-bit address. In addition there is a data bit D and a write bit W.

Design a circuit which output should be the data memorized in the D-Flipflop addressed by $\langle a_1, a_0 \rangle$. In addition if the write bit W is 1, your circuit should write the data from the data bit D to the same D-Flipflop addressed by $\langle a_1, a_0 \rangle$.



Problem 2.47 (Event Detection with RS Flipflops)
Using RS flipflops, you can detect events.

1. Design a sequential logic circuit (draw a graph) with two inputs and two outputs that detects, which out of *two* events occurred first. Use the RS flipflop and elementary gates (AND, OR, NOT, ...). Assume that, initially, all inputs are 0 and the RS flipflop(s) are holding a 0. If input I_i, where $i \in \{1,2\}$, changes its value to 1, output O_i should change its value to 1, and all other outputs should yield 0. The outputs must not change any more when the second input changes to 1.

2. Combine several (how many?) of the circuits from step 1 to a similar event detector for three events.

Note: You need not handle the case of two inputs simultaneously changing to 1.

Problem 2.48 (Binary counters)

In the slides there is an implementation of a D-flipflop with an *enable* input. In practice a different version is more commonly used - the edge-trigerred D-flipflop. Here instead of an *enable* input there is a clock input (*clk*). The difference in operation is that the edge-trigerred D-flipflop only remembers the value of the D input at the one instant when the *clk* input switches from 0 to 1. If *clk* is constantly 0 or constantly 1 the flipflop will not change its state.

Using only such flipflops implement a 3-bit binary counter circuit. The circuit should have only one input 'tick' that will periodically change between 1 and 0. It should have three outputs that count the number of pulses on the input. After the counter counts to 111 it should continue from 000. You can assume the initial state of all flipflops is 0.

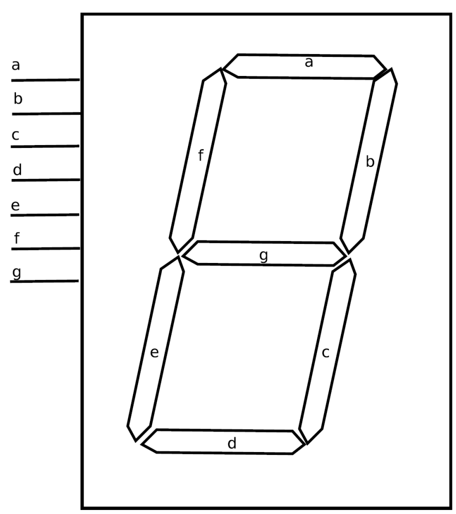
Note: For those of you who are curious here is how an edge-trigerred D-flipflop is built from NAND gates: `http://en.wikipedia.org/wiki/File:Edge_triggered_D_flip-flop.png`. If you're trying to understand this it will help to note that a real physical gate has a certain delay. When the input changes it takes some time (nanoseconds) for the output to react.

Problem 2.49 (Displaying a two-bit number)

Your task for this problem is to create a 2-bit synchronous counter and display the output in a decimal form with the help of 8 light emitting diodes.

You need to assemble this circuit only with the help of the following items:

- 2 positive edge triggered D-flipflops
- 6 NAND gates
- 1 digit display circuit with 8 inputs ($a - g$) corresponding to 8 diodes arranged in the figure below
- 1 signal generator that provides you with a clock signal that you should use to trigger the D-flipflops
- set of wires



Note:

- Basically your task is to create a 2-bit counter and decode the 2-bit output of the counter into 8-bits so that the display shows proper numbers from 0 to 3.
- Positive edge triggered D-Flipflop is just like a normal D-flipflop with the exception that it writes the data when the enable signal (clock) transits from 0 to 1, and in all other cases (constant 0, constant 1, transition 1 → 0) nothing happens.

- You cannot use constant signals.
- For all of the inputs to the 1-digit display logical true (1) means ON and logical false (0) means OFF for the corresponding diode.
- You dont need to worry about the power supplies of the diodes, ICs and the flipflops.

Problem 2.50 (Making a speedometer)

You are working for a car manufacturer and are given the task to make a digital speedometer for a future model. The electrical engineers tell you that they can provide you with two inputs: *rev_tick* very briefly goes from 0 to 1 and then back to 0, whenever the wheels of the car complete one revolution and *ref_clk* that every second very briefly goes form 0 to 1 and then back to 0. You know that the wheels of the car have a circumference of 1 meter. For the initial design you need to provide an electronic circuit that measures the speed in meters per second. You have to provide a number of outputs $a_0, \ldots, a_n$ that represent the current speed. You also know that the car has a maximum speed of 220 km/h.

Imagine that you wanted to display the speed in km/h. What is the maximum resolution your speedometer could achieve? What improvements to the car design can you propose to make this better?

For this problem you should use the edge-trigerred flip-flop together with an extended version that has one additional input R. Whenever R is one, the internal state of the flip-flop is reset to 0 ($Q = 0$) regardless of the state of the D and *clk* inputs. Reseting the internal state when R becomes 1 also happens after a short delay.

2.4 Machines

2.4.1 How to build a Computer (in Principle)

20pt

Problem 2.51 (Hyperpower)
Write an assembler program that reads an integer $n \geq 1$ stored in $P(0)$, and writes n^n in $P(1)$.

10pt

Problem 2.52 (Multiplication)
Write an assembler program (for the assembler language we defined in class) that multiplies the values of data cells 1 and 2 and stores the result in data cell 0.

25pt

Problem 2.53 (Poking zeros)
Given are $n \geq 1$ (n is stored in $P(0)$) integers stored in $P(10) \ldots P(9+n)$, such that no two zeros are next to each other and $P(10) \neq 0 \neq P(9+n)$. Write an assembler program that overwrites all zeros in that array with the sum of the numbers in the neighboring cells of its position.

70pt

Problem 2.54 (Simulating a Register Machine)
Write an SML function `regma` (register machine) that simulates the simple register machine we discussed in class. To represent the program and data store, you should use SML vectors as described in `http://www.standardml.org/Basis/vector.html`. In a nutshell, `Vector.sub(arr,i)` returns the i^{th} element of the vector `arr` and `Vector.update(arr,i,x)` returns the vector `arr`, except that the i^{th} element is replaced by `x`. Finally (useful for testing) `Vector.fromList` makes a vector from a list.

So the the data store should be of type `int vector` and the program store is of type `(instruction * int) vector`, where `instruction` is defined by the following type

```
datatype instruction =
        load | store | add | sub | loadi | addi | subi |
        loadin1 | loadin2 | storein1 | storein2 |
        moveaccin1 | moveaccin2 | movein1acc | movein2acc | movein1in2 | movein2in1 |
        jump | jumpeq | jumpne | jumpless | jumpleq | jumpgeq | jumpmore |
        nop | stop
```

`regma` should take as input a data store `data` and a program store `prog`, and `regma(prog,data)` should return the value of the accumulator register, when the program encounters a `stop` instruction.

20pt
20min

Problem 2.55 (sorting-by-selection)
Let $n \geq 1$ be stored in $P(0)$ and n numbers stored in $P(2) \ldots P(n+1)$. Write an assembler program that performs a sorting by selection and outputs the result in $P(n+2) \ldots P(2n+1)$. Write comments to each line of your code (like in the example codes from the slides). Uncommented code will not be considered.

20pt
20min

Problem 2.56 (Binary to decimal)
Let $P(0) = n$ contain the number of bits of a binary number stored in $P(2) ... P(2+n-1)$. Each memory cell represents one bit of the number where $P(2)$ is the least significant bit and $P(2+n-1)$ is the most significant bit. Write a program that stores the corresponding decimal number in $P(1)$.

2.4.2 A Stack-based Virtual Machine

Problem 2.57 (Reasons for Virtual Machines)
Thinking back to the lectures about $\mathcal{L}(\texttt{VM})$ and `SW`, sum up the benefits of compiling programs in high-level languages to the language of a virtual machine instead of directly compiling them to an assembler language `ASM`.

12pt
15min

Problem 2.58 (Binary Conversion in $\mathcal{L}(\texttt{VM})$)
Write a $\mathcal{L}(\texttt{VM})$ program that converts a binary natural number into a decimal natural number. Suppose that n, the number of digits, is stored in `stack[2]` and n numbers 0 or 1 above it follow, where the top of stack is the least significant bit. `stack[0]` and `stack[1]` are available for your

use. Your program should leave only the converted number on the stack (in `stack[0]`). You are allowed to use labels for (conditional) jumps.

For instance an initial stack $\begin{array}{|c|}\hline 1 \\ \hline 0 \\ \hline 1 \\ \hline 3 \\ \hline ? \\ \hline ? \\ \hline \end{array}$ should give the result stack $\boxed{5}$.

12pt

Problem 2.59 (Fibonacci Numbers)

Assume the data stack initialized with `con` n for some natural number n. Write a $\mathcal{L}(\texttt{VM})$ program that computes the n^{th} Fibonacci number and returns it on the top of the stack.

Hint: Remember that the n^{th} Fibonacci number is given by the following recursive equations:

$$fib(n+1) = f(n) + fib(n-1) \qquad fib(0) = 0 \qquad fib(1) = 1$$

2.4.3 A Simple Imperative Language

20pt

Problem 2.60 (Convert Highlevel Code to VM Code)

Given is an array `A[0..10]` and the following piece of imperative code:

```
for j := 1 to 5 do
 for i := j to 10-j do
  A[i] := A[i-j] + A[i+j];
```

Suppose the array is loaded on stack (top value being `A[10]`). Convert the code into VM code.

Problem 2.61 (Static procedure for logarithm)

Write down a static procedure in $\mathcal{L}(\texttt{VM})$ that computes $f(x) = \lfloor \log_2(x) \rfloor$. This procedure should not be recursive. Use the new `lpeek` and `lpoke` instructions from the previous exercise. Is there something you do at the end of your procedure that is not part of your algorithm. If yes, then describe a more elegant way of doing that by modifying the behavior of an existing `VM` instruction.

Hint: Remember that at the end of a static procedure call exactly one value - the result - should be left on the stack.

6pt
10min

Problem 2.62 (While Loop in $\mathcal{L}(\texttt{VM})$)

Write a program in the Simple While language that takes two numbers A and B, given at the memory addresses 1 and 2, and returns $(A+B)^{42}$. Show how the compiled version of it looks like in the Virtual Machine Language $\mathcal{L}(\texttt{VM})$ (concrete, not abstract syntax).

Problem 2.63 (Simple While program on Fibonacci)

Write a Simple While Program that takes a number N and computes the N^{th} Fibonacci number. Then provide the Abstract Syntax for your code.

Show how the $\mathcal{L}$(VM) version of it looks like by compiling it.

Hint: Remember that the n^{th} Fibonacci number is given by the following recursive equations:

$$fib(n+1) = f(n) + fib(n-1) \qquad fib(0) = 0 \qquad fib(1) = 1$$

2.4.4 Compiling Basic Functional Programs

Problem 2.64 (Cross identifiers?)

Now suppose you want to compile a μML program containing a few function declarations such that they use the local identifiers from the functions defined above. For example,

```
([("F1", ["n","a"], Sub(Id "n",Id "a")),
  ("F2", ["m","x"], Mul(Add(Id "m",Id "x"),Id "n"))],
 App("F2", [Con 1, Con 2]) );
```

Will such a program compile? If yes, will it execute correctly? Explain your answer.

Hint: You may want to track down the compilation process on a given example.

Problem 2.65 (Duplicate identifiers?)
Suppose you want to compile a μML program containing a few function declarations such that some of them contain the same identifier names such as

```
([("F1", ["n","a"], Sub(Id "n",Id "a")),
 ("F2", ["m","n","a"], Mul(Add(Id "m",Id "a"),Id "n"))],
 App("F2", [Con 1, Con 2, Con 3]) );
```

Will such a program compile? If yes, will it execute correctly? Explain your answer.

Hint: You may want to track down the compilation process on a given example.

Problem 2.66 (Prime numbers)
Write a program in μML that takes an integer $n > 1$ and returns 1 if the number is prime and 0 otherwise. Your program should be a pair of a well defined list of function declarations, and a single `App` call of the main function. Obviously, that function will call the helping function(s) in its body and helping functions may call themselves. Can you solve the problem using only two helping functions?

2.4.5 A theoretical View on Computation

Problem 2.67: Explain the concept of a Turing machine, what is it used for? What is a universal Turing machine? 10pt 10min

Problem 2.68 (Turing Machine) 12pt
Given the alphabet $\{0, 1\}$ and a initial tape that starts with 0,1,0.

1. Define a transition table that converts the three entries of this tape to 1,0,1 and terminates afterwards independently of the tape's tail.

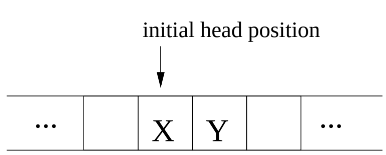
2. Give an example initial tape where your transition table wouldn't terminate or argue why such an initial tape can't exist.

Hint: The Turing machine terminates when there is no action in the transition table applicable.

Problem 2.69 (Boolean And) 5pt 8min
Suppose a tape with only two cells arbitrarily filled with 0 or 1 and the head of the Turing machine over the left cell. Define a transition table such that the machine always terminates with a final state where the left cell has value 1 if and only if both cells contained 1 in the initial state; i.e. the machine should evaluate the a boolean "and".

Hint: Admissible moves are *left*, *right*, and *stop* with the obvious meaning.

Problem 2.70 (Boolean Equivalence) 11pt 20min
Consider a tape arbitrarily filled with ones and zeros and the head initially positioned over some cell "X" as depicted below



Define a transition table for an always terminating Turing machine TM that computes the boolean equivalence of “X” and “Y”: Upon halting, your TM should return the value 1 in cell “X” if the values of the cells “X” and “Y” were initially equal and otherwise 0.

Try to use as few states as possible. The number of points you can obtain for this exercise is $\max(0, 14 - x)$, where x is the number of states of your working TM.

Hint: You only need to consider the two cells “X” and “Y”. It does not matter where the head stays when the TM terminates.

Note:

1. Admissible moves are *left*, *right*, and *none* with the obvious meaning.
2. You are free to overwrite the initial value of “Y” and to introduce additional symbols in the alphabet, if you need it for your solution.

Problem 2.71 (Halting Reductions)

The fact that a TM cannot decide if another TM halts on a given input is not the only limit of computation. There are a lot of other things TM’s cannot do, and the halting problem can be used to prove this. This process is called ”reduction to the halting problem”: for proving that a TM cannot decide a certain a property P, assume that it could and then use it to construct another TM that can decide the halting problem (i.e. to decide if some TM halts on some given input).

For the following statements, provide a proof by reduction to the halting problem or a counterexample:

- No TM can decide in general whether another TM halts on all inputs.
- TM can decide in general whether another TM uses all its states in the computation on a given input x.

Hint: Here is an example of how to solve such a task. All you need to do is to figure out how to adapt this to the points above.

- Prove or refute that no TM can decide in general if another TM halts on the empty input.
- Assume we have a machine M that can decide if another TM halts on the empty input. We want to decide if a given TM N halts on input x. We can construct a machine K that started on the empty input, writes x on the tape and then simulates $N(x)$. If $M(K)$ (M run on a coded version of K as input) outputs yes, then it means that K halted on the empty input, thus N halted on x, no means the opposite. Thus, we can decide the halting problem, which is false.

Problem 2.72 (Number of Steps of a Turing Machine)

Let $s_{\max}(n)$ be the maximum number of steps that an n-state Turing machine with the alphabet $\{0, 1\}$ can take on an empty tape, halting in the end. Is the function $s_{\max}$ computable? Give a proof or a refutation.

Hint: If we had an implementation of $s_{\max}$, how could we implement the `will_halt` function from the lecture using $s_{\max}$?

Note: From the lecture, we know that it is impossible to implement a function `will_halt(program, input)`. Assume the following corollary, known as the “halting problem on the empty tape”, as given: It is even impossible to write a Turing machine (or an equivalent function `will_halt_empty(program)`, resp.) that tells whether an arbitrary Turing machine halts on an *empty* tape.

Problem 2.73 (TM and languages)

Design a Turing Machine which accepts the language $\{101100...1^{n-1}0^{n-1}1^n0^n \mid n > 0\}$ (halts with ”yes” if such input is given and halts with ”no” otherwise). First describe in plain English the core idea of how your algorithm works. Think of possible wrong inputs, and show how your TM handles them.

Note:

- The point of this exercise is to help you think of how to approach and solve a problem. Imagine you are given 0 points for a TM which only partially works (some wrong inputs can pass as accepted or the other way around).

- For exercises about TM construction, please format the transition table according to the TM simulator at `http://ironphoenix.org/tril/tm/` (here you will also find some example programs). This way you will be able to check your "code" and your TAs will have an easier time grading.

Problem 2.74 (TM and TCN numbers)
Given a tape with an n-bit binary number written after symbol $+$ or $-$ (denoting if the number is positive or negative), design a Turing Machine which will convert it to a TCN. Initially, the head is over the sign symbol. There is no restriction where would the head be after halting. If the number of states exceeds 4, you will lose 2 points per extra state. **Uncommented code will not be graded.**

For example we would have

```
Input: -101
Output: 1011
```

Problem 2.75 (Turing Machine Simulating a Half Adder)
Given the alphabet $\{0, 1\}$ and a finite set of states of your choice. Define upon these sets a transition table that behaves like a half adder, i.e. it reads two bits from the tape and writes a sum and carry bit on the tape again (at any arbitrary but fixed position).

2.5 The Information and Software Architecture of the Internet and WWW

2.5.1 Overview

nothing here yet

2.5.2 Internet Basics

nothing here yet

2.5.3 Basics Concepts of the World Wide Web

Problem 2.76 (Quiz for the TAs)

Your last assignment this semester is to give your TAs a quiz. We hope you will enjoy this :)

You need to create a form in HTML that contains the following:

1. Include at least 5 multiple choice questions.
2. All following concepts: button, radio button, check box, drop down box, text input.
3. At least one image and one working link.
4. Tables, lists.
5. Make it look nice overall (styles, colors ...)

You can provide a fictive action attribute.

Hint: HTML is useful and easy to learn. Start by finding a nice tutorial online.

Problem 2.77 (HTML basics)
Answer the following questions about HTML:

1. What does HTML stand for?
2. Who is making the Web standards?
3. What is HTML tag for the largest heading?

4. What is the correct HTML tag for inserting a line break?
5. What is the correct HTML for adding a background color?
6. What is the correct HTML tag to make a text bold?
7. What is the correct HTML tag to make a text italic?
8. What is the correct HTML for creating a hyperlink?
9. How can you create an e-mail link?
10. How can you open a link in a new browser window?
11. Which of these tags are all `<table>` tags?
    - `<thead><body><tr>`
    - `<table><head><tfoot>`
    - `<table><tr><tt>`
    - `<table><tr><td>`
12. What is the correct HTML to left-align the content inside a tablecell?
13. How can you make a list that lists the items with numbers?
14. How can you make a list that lists the items with bullets?
15. What is the correct HTML for making a checkbox?
16. What is the correct HTML for making a text input field?
17. What is the correct HTML for making a drop-down list?
18. What is the correct HTML for making a text area?
19. What is the correct HTML for inserting an image?
20. What is the correct HTML for inserting a background image?

Problem 2.78 (For Future Generations)

As one of the last assignments, we would like you to look a bit into the future. Imagine yourselves one year from now. Some of you will definitely be TAs at that time, so it's time to show your creativity and teaching skills. Your task is to basically create an HTML form representing the examination you would give to the freshmen in 2012. It can be any midterm or final for GenCS I or II. There are only a few specifications you must look out for. The rest is fully up to you.

The web form must:

1. Include multiple choice and 'fill in the blanks' questions, enough for an actual exam time of 75 or 120 minutes.
2. Include all of the following: button, radio button, check box, drop down box, text input.
3. The exam must contain figures and sections of code from any of the studied programming languages that you ask questions on.
4. Link your exam to some useful pages. Make it like an 'open book' exam and offer some actual existing resources.
5. The overall style should be professional. Put a bit of effort into appearance and aesthetics.

6. In the end, the scoring system should work. Nothing too fancy, but it should be an operational exam from start to finish.

Hint: HTML is useful and easy to learn. Start by finding a nice tutorial online. You might wish to consider JavaScript for your scoring mechanism. Also, CSS is recommended for brushing up your design!

Problem 2.79 (Web browsers)

- What is the difference between a web page and a web site?
- What is a web browser? Name at least 5 practical web browser tools.

2.5.4 Web Applications

Nothing here yet

2.5.5 Introduction to Web Search

Problem 2.80 (SML Web Crawler)
A web crawler is a program that will store a copy (mirror) of a web site. Generally, crawlers access a given web page and, after retrieving the HTML source, they extract the links and also download those pages (or images or scripts). This will provide the user the possibility to access these pages even when they are not connected to the internet or to perform different measurements on the pages.

Your task is to write your own SML Web Crawler, following these steps:

1. Make sure that you downloaded and understood the SML sockets example file used in the last assignment. Use the following updated socketReceive function:

   ```
   (* Receives maxbytes bytes from the socket. Returns the string message. *)
   fun socketReceive(sock, maxbytes) =
          Byte.bytesToString(
                 Socket.recvVecNB(sock, maxbytes)
          );
   ```

   The problem with this function is that, if the server sends a message longer than maxbytes, all the remaining bytes will be queued on the socket, but not processed. Write your own `fullMessage` function that overcomes this problem by reading the whole reply from the server (you can use `socketReceive`, it will return a string of length 0 if the message from the server is finished). Your function should have the following type:

   ```
   val fullMessage= fn : ('a,Socket.active Socket.stream) Socket.sock -> string
   ```

2. Now, write a method that, given a *host* and *page*, will make a HTTP **GET** request to the server for the given *page* on that *host*, and will return the HTTP response. Your function should have the following signature:

   ```
   val getPage = fn : string * string -> string
   ```

   For example, you should be able to run `getPage("en.wikipedia.org","/wiki/Main_Page")` and retrieve the home page of Wikipedia.

Hint: Try to do the request on telnet first, by connecting to the *host* on port 80. Check resources online (i.e. Wikipedia) on how to make a valid HTTP request.

---

3. Now that you have the HTTP response, check it closely and you will discover that it contains the HTML web page, but also some headers. In order to be sure that you will only store the HTML page, write a function `extractHTML` that scans the string and discards everything that **is not** between `<html>` and `</html>`. Of course, your function will have the signature:

```
val extractHTML : string -> string
- extractHTML("Discard me! <html><head><title>Hello!</title></head></html>");
val it = "<html><head><title>Hello!</title></head></html>" : string;
```

4. Write a function `extractLinks` that will go through your HTML source code and will return all the links that it contains. Feel free to look into the `HTML` or `RegExp` library of SML, but making your function only going through the string and extracting sequences like the following will suffice:

   ```
   <a href="extract me!">...
   <img src="extract me!"> ...
   ```

   You are not required to handle links other than the ones found in anchors and images. Your function will have the following signature (get a string and return a list of strings which are the links found):

   ```
   val extractLinks = fn : string -> string list;
   ```

5. Mind the fact that these links might contain the protocol ("http://"), might be relative to the root of the host ("/img/happy.png"), or might be relative to the current page ("next/index.html"). Your `getPage` function requires a *host* and a *page* as arguments, and the *page* should be relative to the host root (i.e. absolute path). Write an SML function `normalizeLinks` that, given a host, page and list of strings, will return a list of pairs $(host, page)$ that can be used by the `getPage`:

   ```
   val normalizeLinks : string * string * string list -> (string * string) list;
   normalizeLinks("www.example.com", "/en/test.html",
          ["http://www.google.com/something/x", "/img/happy.png", "next/index.html"]
   );
   val it = [
          ("www.google.com", "/something/x"),
          ("www.example.com", "/img/happy.png"),
          ("www.example.com", "/en/next/index.html")
   ] : (string * string) list;
   ```

6. This sub-task will be to write the wrapping crawler function.

   Have a look at the following SML function that writes a string to a file:

   ```
   fun writeToFile(file, content) =
          let
                 val os = TextIO.openOut(file)
                 val vc = String.toString(content) (* we need an SML vector *)
                 val _ = TextIO.output(os, vc)
                 val _ = TextIO.flushOut(os)
          in
                 TextIO.closeOut(os)
          end;
   ```

   Hint: You might want to extend this function to also handle folders, such that you can store the pages or images relative to the root page you start your crawl on. However, you are not requested to do so.

   This function will be used in storing the HTML page to disk. Your crawler will have the following signature:

   ```
   val crawler : string * string * int -> unit;
   ```

   The first two parameters are the host and the starting page (i.e. "`www.example.com`" and "`/test/index.html`"). The third parameter is an integer representing the maximum depth you should go into. You will follow the following steps:

(a) use `getPage` to retrieve the HTTP response

(b) use `extractHTML` to extract only the HTML part of the response

(c) write the HTML part to a file (see the note below!)

(d) use `extractLinks` and normalizeLinks to get the list of links to follow further

(e) recursively call the `crawler` method; remember to decrease the depth and not proceed with a negative depth!

Note: There might be problems with storing images. We will not grade this problem based on the output, but rather on how well you managed to follow the instructions and on your intermediary results. Please think about what the problem with images is and write a short comment at the end of your sml file!

Problem 2.81 (Ranking pages)
In this task you will gain some practical experience with a real-world web crawler and you will come up with your own page ranking procedure!

Look into the man pages of `wget` (available on `linux`, use the tlab machines if you don't have `linux` already on your laptop; you might also find `Windows` ports of the program). `wget` has the ability to follow links while saving the pages to disk, and also to keep the directory structure consistent with the server.

Choose a web page of your preference (we recommend using a wikipedia page) and run `wget` with a depth limit of your choice. Now inspect the output directory and observe items that might help you in ranking your web pages (for example, number of links pointing to a web page, number of images, length of the content or its age might be starting points!). Do not reinvent the wheel, or reverse-engineer the Google PageRank algorithm! Be creative and make a good use of the features that your starting page has (wikipedia has, for example, the links between related topics). Also, do not take into consideration whether the features are (easily) computable.

You will have to supply a PDF document reporting your actions. Describe how you used wget to mirror the site (do include the commands used!). Describe your ranking function (what items you consider, how they influence the page score). Compile a table which contain these items, the score of each item for each page and the final score of the page.

Finally, write down your observations and comments about the method that you employed.

2.5.6 Security by Encryption

Nothing here yet

2.5.7 An Overview over XML Technologies

Nothing here yet.

2.5.8 The Semantic Web

nothing here yet

2.6 Legal Foundations of Information Technology

2.6.1 Intellectual Property, Copyright, and Licensing

nothing here yet

2.6.2 Information Privacy

nothing here yet

3 Search and Declarative Computation

3.1 Problem Solving and Search

3.1.1 Problem Solving

[1] EdN:1

Problem 3.1 **(Sudoku)**

					8			4
	8	4		1	6			
			5			1		
1		3	8			9		
6		8				4		3
		2			9	5		1
		7			2			
			7	8		2	6	
2			3					

This question will give you an excuse to play Sudoku (see www.websudoku.com for explanation) while doing homework. Consider using search to solve Sudoku puzzles: You are given a partially filled grid to start, and already know there is an answer.

- Define a state representation for Sudoku answer search. A state is a partially filled, valid grid in which no rows, column, or 3x3 square contains duplicated digits. Also specify what transitions would be.
- If the puzzle begins with 28 digits filled, what is L, the length of the shortest path to goal using your representation?
- On a typical PC, which search algorithm would you choose: BFS, DFS or IDS? Why?

Problem 3.2 **(Define Problem Formulation)**
Define the concept of *Problem Formulation.*

Problem 3.3: Does a finite state space always lead to a finite search tree? How about a finite space state that is a tree? Justify your answers.

Problem 3.4 **(Problem formulation)**
You and your roommate just bought an 8 liter jug full of beer. In addition you have two smaller empty jugs that can hold 5 and 3 liters respectively. Being good friends you want to share the beer equally. For this you need to split the amount in two separate jugs and each should contain exactly 4 liters. Write a formal description of this problem. What is one possible solution? What is the cost of your solution?

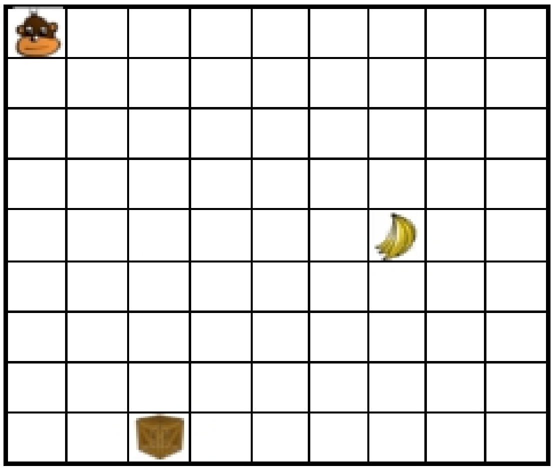
Problem 3.5 **(Problem formulation and solution)** 20pt 20min

a) Write a problem formulation and path cost for each of the following problems:

   1. A monkey is in a room with a crate, with bananas suspended just out of reach on the ceiling. The monkey would like to get the bananas.
   2. You have to color a complex planar map using only four colors, with no two adjacent regions to have the same color.

[1] EdNote: we should extract some problem formulation sub-problems from e.g. moving-knight

b) Given the following concrete examples of the two problems from (a), provide a solution for each of the examples that conforms the problem formulation you gave in (a) and specify the cost of this solution according to the path cost you defined.



Hint: Refer to the slides for specifications regarding problem formulation and solution. Path cost is a function that assigns cost to every operator.

Problem 3.6 (Search of the max element) 8pt 10min

Formalize the task of finding the maximum element in a set of the integer numbers. What are the properties of your search? Justify your answers.

3.1.2 Search

2 EdN:2 20min

Problem 3.7 (The Dog/Chicken/Grain Problem) 20pt

A farmer wants to cross a river with a dog, a chicken, and a sack of grain. He has a boat which can hold himself and either of these three items. He must avoid that either dog and chicken or chicken and grain are together alone on one river bank, since otherwise something gets eaten.

1. Represent the farmer's problem of crossing the river without losing his goods as a search problem.

2. Draw a sufficiently large portion of the search tree induced by this problem to exhibit a solution.

3. Discuss three search strategies and their advantages and disadvantages in this scenario.

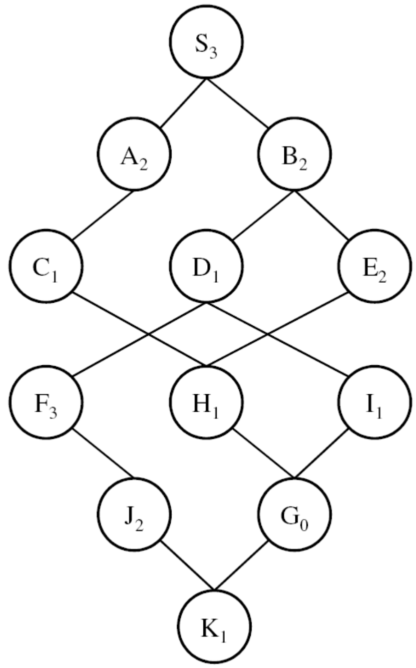
Hint: The farmer can also take something back over the river.

Problem 3.8 (Moving a Knight)

Consider the problem of moving a knight on a 3x4 board, with start and goal states labeled as S and G in the figure below. The search space can be translated into the following graph. The letter in each node is its name and you do not need to worry about its subscript for now.

[2] EdNote: need to take these problems apart, so that they do not mention specific search strategies

S_3	H_1	D_1	K_1
I_1	J_2	A_2	E_2
C_1	B_2	G_0	F_3



Make the following assumptions:

- The algorithms do not go into infinite loops (i.e. once a node appears on a path, it will not be considered again on this path)
- Nodes are selected in alphabetical order when the algorithm finds a tie.

Write the sequence of nodes in the order visited by the specified methods (until the goal is reached). Note: You may find it useful to draw the search tree corresponding to the graph above.

- DFS
- BFS

3.1.3 Uninformed Search Strategies

Problem 3.9 (Uninformed Search)
Explain all uninformed search strategies introduced in class and compare their advantages and disadvantages with respect to completeness, time, space, and optimality.

[3] EdN:3

Problem 3.10 (Sudoku)

[3]EdNote: we need to take the sudoku problem apart and only have the third bullet point here

					8			4
	8	4		1	6			
			5			1		
1		3	8			9		
6		8				4		3
		2			9	5		1
		7			2			
			7	8		2	6	
2			3					

This question will give you an excuse to play Sudoku (see www.websudoku.com for explanation) while doing homework. Consider using search to solve Sudoku puzzles: You are given a partially filled grid to start, and already know there is an answer.

- Define a state representation for Sudoku answer search. A state is a partially filled, valid grid in which no rows, column, or 3x3 square contains duplicated digits. Also specify what transitions would be.
- If the puzzle begins with 28 digits filled, what is L, the length of the shortest path to goal using your representation?
- On a typical PC, which search algorithm would you choose: BFS, DFS or IDS? Why?

Problem 3.11: Describe a state space in which iterative deepening search performs much worse than depth-first search (for example $O(n^2)$ vs. $O(n)$).

Problem 3.12 (Actions with Negative Costs)
Suppose that actions can have arbitrary large negative costs.

1. Explain why this possibility would force any optimal algorithm to explore the entire state space.
2. Does it help if we insist that step costs must be greater than or equal than to some negative constant c? Justify your answer.

Problem 3.13 (Implementing Search) 50pt
Implement the depth-first and breadth-first search algorithms in SML. The functions `depthFirst` and `breadthFirst` take three arguments that make up the problem description:

1. the initial state
2. a function `next` that given a state `x` in the state tree returns at set of pairs `(action,state)`: the next states (i.e. the child nodes in the search tree) together with the actions that reach them.
3. a predicate (i.e. a function that returns a Boolean value) `goal` that returns `true` if a state is a goal state and `false` else.

the result of the functions should be the goal state together with a list of actions that reaches the goal state from the initial state.

Hint:

1. Write an auxiliary function that takes the fringe (i.e. a list of unexpanded states together with the plans to reach them) as an accumulator argument.
2. It is always good to treat the failure case with an exception.

3. The problem may become simpler to think about, if you first write a function that does not care about actions, which makes `next` simpler and also the return actions of the auxiliary function.

Problem 3.14 (Implementing Search)
Implement the depth-first and breadth-first search algorithms in SML. The corresponding functions `dfs` and `bfs` take three arguments that make up the problem description:

1. the initial state
2. a function `next` that given a state `x` in the state tree returns at set of pairs `(action,state)`: the next states (i.e. the child nodes in the search tree) together with the actions that reach them.
3. a predicate (i.e. a function that returns a Boolean value) `goal` that returns `true` if a state is a goal state and `false` else.

The result of the functions should be a pair of two elements:

- a list of actions that reaches the goal state from the initial state
- the goal state

The signatures of the two functions should be:

```
dfs : 'a -> ('a -> ('b * 'a) list) -> ('a -> bool) -> 'b list * 'a
bfs : 'a -> ('a -> ('b * 'a) list) -> ('a -> bool) -> 'b list * 'a
```

where `'a` is the type of states and `'b` is the type of actions.
In case of an error or no solution found raise an `InvalidSearch` exception.

Hint:

1. Write an auxiliary function that takes the fringe (i.e. a list of unexpanded states together with the plans to reach them) as an accumulator argument.
2. It is always good to treat the failure case with an exception.
3. The problem may become simpler to think about, if you first write a function that does not care about actions, which makes `next` simpler and also the return actions of the auxiliary function.

---

[4] EdN:4

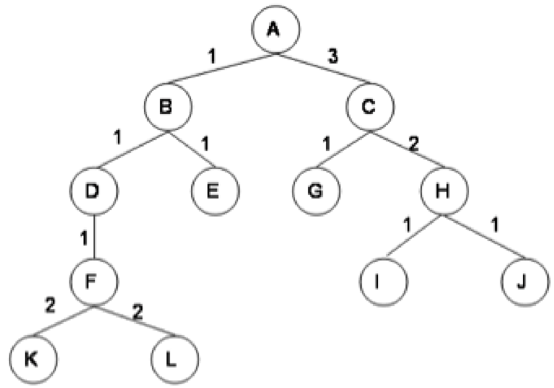
Problem 3.15 (A Trip Through Romania) 30pt
Represent the Romanian map we talked about in class in a concrete `next` function. Search with the procedures from Problem 3.13 a trip from Arad to Bucharest. Compare the solution paths and run times.

Problem 3.16 (Relations between search strategies) 15pt
Prove or refute each of the following statements:

1. Breadth-first search is a special case of uniform-cost search.
2. Breadth-first search, depth-first search, and uniform-cost search are special cases of best first searches.

Problem 3.17 (Search Strategy Comparison on Tree Search)
Consider the tree shown below. The numbers on the arcs are the arc lengths.

[4] EdNote: make a separate problem in formaulation from the problem representation in SML and reference this here.



Assume that the nodes are expanded in alphabetical order when no other order is specified by the search, and that the goal is state G. No visited or expanded lists are used. What order would the states be expanded by each type of search? Stop when you expand G. Write only the sequence of states expanded by each search.

Search Type	Sequence of States
Breadth First	
Depth First	
Iterative Deepening (step size 1)	
Uniform Cost	

Problem 3.18 (Missionaries and cannibals)
Three missionaries and three cannibals are on one side of a river, along with a boat that can hold one or two people. The final goal is to get everyone to the other side, without ever leaving a group of missionaries in one place outnumbered by the cannibals in that place.

1. Formulate the problem precisely. When defining the operators, it is not necessary that you write every possible *state* → *state* combination, but you should make it clear how one would derive the next state from the current one.

2. Suppose the next-function for depth first search (DFS) and breadth first search (BFS) expands a state to its successor states using the operators you have defined in 1. in the order you have defined them. Operators that leave more cannibals than missionaries on one side will not be considered. Likewise, operators that lead to the immediate previous state will not be considered (e.g., after moving a cannibal from left to right, the next-function for this state will not include a state where a cannibal moves from right to left). Draw the search tree till depth 3. What are the first 5 nodes explored by DFS? What are the first 5 nodes explored by BFS?

3. If you would implement this problem, would you rather use BFS or DFS to find the solution? Briefly explain why?

50pt

Problem 3.19: Write the `next` function, `goal` predicate and `initial_state` variable for the 8-puzzle presented on the slides (please check the slides for the description). Then use these to test your breadth-first and depth-first search algorithms from the previous problem.

Use the following :

```
datatype action = left|right|up|down;
type state = int list;(*9 elements, in order, 0 for the empty cell*)
```

Refer to the slides for the `initial_state` variable. Make sure that if an action is illegal for a certain state, it does not appear in the output of next.

Sample testcase:

```
test call : next(initial_state);

output: [(left,[7,2,4,0,5,6,8,3,1]),(right,[7,2,4,5,6,0,8,3,1]),
   (up,[7,0,4,5,2,6,8,3,1]),(down,[7,2,4,5,3,6,8,0,1])];
```

15pt

Problem 3.20 (Interpreting Search Results)

The state of Ingushetia has only four cities (A, B, C, and D) and a few two-way roads between them, so that it can be modeled as an undirected graph with four nodes. The task is to go from city A to city D. The UCS algorithm finds a solution to this task that is 10km shorter than the one BFS finds. The solution of BFS in turn is 10km shorter than the one of the DFS algorithm.

Draw a map of Ingushetia with roads and their distances that satisfies both conditions. What paths between A and D in your map will be found as solutions by each of those algorithms?

Note: All algorithms had repetition checking implemented, so that when a node is expanded, all its children that belong to a list of previously expanded nodes during the execution of that algorihtm are ignored. In addition, when no order of choosing a node for expansion is specified by an algorithm, expansion in alphabetical order takes place.

14pt

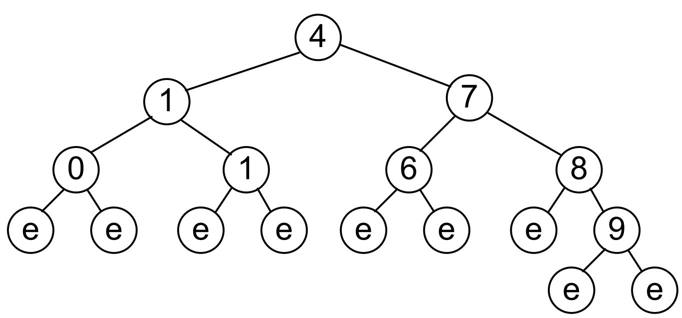
Problem 3.21 (Treesort Function)

Your task is to write a treesort function in SML that sorts a list of integers by first creating a binary search tree from the list and then loading the tree (in a sorted order) back into a list.

Use the following definition of a binary search tree:

- All leaves are empty nodes.
- All internal nodes carry a value and a left and a right subtree.
- The values of all nodes in a node's left subtree are smaller than the node's value and all nodes in its right subtree are greater or equal to the node's value.

The following tree is an example of a binary search tree:



Given the following datatype:

```
datatype searchtree = empty | node of searchtree*searchtree*int;
```

The tree above would be represented as follows:

```
node(node(node(empty,empty,0),node(empty,empty,1),1),
node(node(empty,empty,6), node(empty,node(empty,empty,9),8),7) , 4);
```

Write the functions using the searchtree datatype. The function sort should be of the following type:

```
fn treesort: int list -> int list
```

Problem 3.22 **(Power Source Search)**
A robot is on the 5x5 map shown below. It wants to reach a power source, but its sensors only allow it to detect the source once it is in the same cell with it. Find a problem formulation in the quadruple format presented in the lecture such that depth first search will find a solution after expanding exactly 6 nodes.

Assume that the `next` function of the DFS algorithm used returns the `(action, state)` tuples in the order in which the corresponding operators are defined. For example, if your operators are `jump` and `sing`, then the next function called on state i would return a list `[(jump, state j), (sing, state k)]` and not the other way around. (this is just an example, these operators will not do a very good job ... :))

Define a path cost for this problem. What is the cost of this solution? Is the solution optimal?

How many node expansions would BFS make considering the same `next` function?

			R	
P				P

R represents the robot and P a power source.

[5] EdN:5

Problem 3.23 **(Maximum independent set)**

An independent set of vertices in a graph G is a set where no two vertices are adjacent. The maximum independent set of vertices in a graph is an independent set with the greatest number of vertices. This number is denoted as $\alpha(G)$.

- Using what we have learned about search, how can you construct a representation that can be used to find a maximum independent set in a graph?
- What search algorithm is most appropriate?
- Estimate the number of maximum independent sets in a graph

3.1.4 Informed Search Strategies

Problem 3.24 **(A looping greedy search)**
Draw a graph and give a heuristic so that a greedy search for a path from a node A to a node B gets stuck in a loop. Draw the development of the search tree, starting from A, until one node is visited for the second time.

Indicate, in one or two sentences, how the search algorithm could be modified or changed in order to solve the problem without getting stuck in a loop.

Problem 3.25 **(A^* Theory)**
What is the condition on the heuristic function that makes A^* optimal? Does a heuristic with this condition always exist? 10pt

Problem 3.26 **(A variant of A^*)**
Imagine an algorithm B^* that uses the evaluation function $f(n) = g(n) \cdot h(n)$, where $g(n)$ is the path cost to the current node n, and $h(n)$ is a heuristic function. Is this algorithm better or worse than A^*? Explain your findings. What does $h(n)$ represent?

Problem 3.27 **(True or False on A^*)**
True or False? Explain why.

---

[5] EdNote: take the next problem apart as well.

1. A^* search always expands fewer nodes than DFS does.

2. For any search space, there is always and admissible and monotone A^* heuristic.

Problem 3.28 **(Sudoku Revisited)**

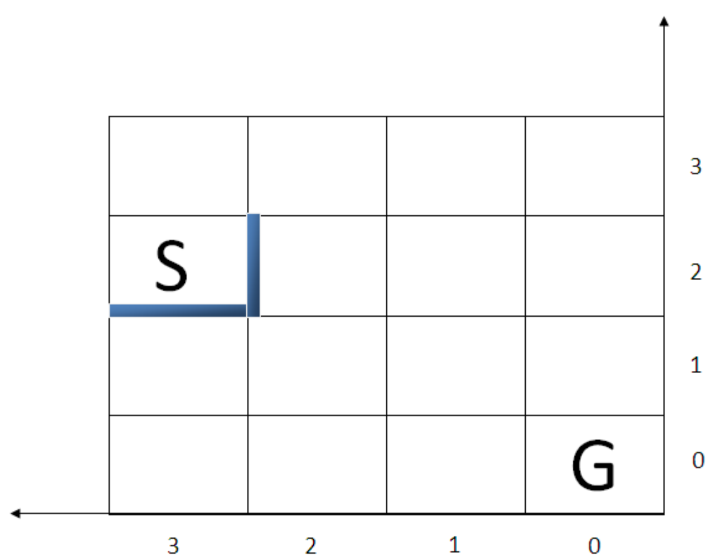
					8			4
	8	4		1	6			
			5			1		
1		3	8			9		
6		8				4		3
		2			9	5		1
		7			2			
			7	8		2	6	
2			3					

Remember the Sudoku problem from the last homework. You were asked which search algorithm you would choose on a typical PC: BFS, DFS or IDS. Is A^* better than your first choice? What is an admissible heuristic for A^*?

12pt
15min

Problem 3.29 **(Monotone heuristics)**
Let $c(n, a, n')$ be the cost for a step from node n to a successor node n' for an action a. A heuristic h is called *monotone* if $h(n) \leq h(n') + c(n, a, n')$. Prove or refute that if a heuristic is monotone, it must be admissible. Construct a search problem and a heuristic that is admissible but not monotone. Note: For the goal node g it holds $h(g) = 0$. Moreover we require that the goal must be reachable and that $h(n) \geq 0$.

Problem 3.30 **(A Good Old Friend, the Maze)**



Given a maze like the one above, consider using search to find the way from start to goal. The shaded areas are walls. You start from S and can only go left, right, up or down (unless there is a wall). All movements cost the same. The heuristic function is the Manhattan distance,

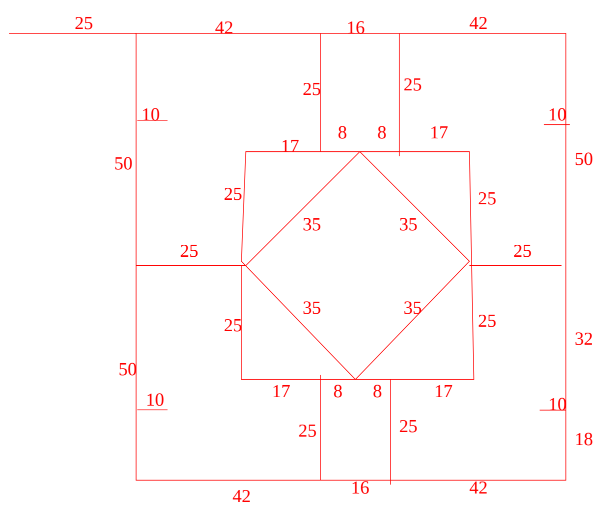
$h = |x_1 - x_2| + |y_1 - y_2|$. For the following questions, explanations are required (simple answer is not enough).

1. Is this an admissible heuristic for A^* for the maze problem?
2. Is it an admissible heuristic if you can move in 8 directions instead of 4 (so also diagonally), if any movement still costs the same?
3. Which performs better with this heuristic, A^* or simple Greedy?
4. For the case of moving in all 8 directions, is the Euclidean distance, $h_e = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$, admissible?
5. For the case of moving in all 8 directions, provide an admissible heuristic that is different from h and h_e, call it h_1, such that h_1 is non-trivial (non-constant and not the hardcoded actual cost).
6. Getting back to the 4 direction movement, is h_e more efficient for A^* than h?

Problem 3.31 **(A^* search on Jacobs campus)** 45pt

Implement the A^* search algorithm in SML and test it on the problem of walking from the main gate to the entrance of Research 3 with linear distance as heuristic. The length of line segments are annotated in the map below.

No function signature is provided, instead at the end of your program call your function so that it prints the actions needed to reach the entrance and the associated cost.



Problem 3.32 (Relaxed Problem)
The relaxed version of a search problem P is a problem P' with the same states as P, such that any solution of P is also a solution of P'. More precisely, if s' is a successor of s in P, it is also a successor in P' with the same cost. Prove or refute that for any state s, the cost $c'(s)$ of the optimal path between s and the goal in P' is an admissible A^* heuristic for P.

Hint: Think about the graphical representation of the problems.

Problem 3.33 (Relations between search strategies) 15pt
Uniform-cost search is a special case of A^* search.

Problem 3.34 (Global Solutions)
For each of the following algorithms, briefly state why or why not they are guaranteed to converge to a global optimum on a problem P:

1. A^* search with the heuristic from the problem above
2. Greedy search with the same heuristic
3. Hill Climbing
4. Genetic Algorithms

3.1.5 Local Search

Problem 3.35 (Local Search)
What is a *local* search algorithm?

1. What does the "fringe" known from generic search algorithms look like in a local search algorithm?
2. What is the space complexity of local search?
3. Name two practical applications for local search.
4. Name a simple algorithm for local search. Give a brief overview of its advantages and disadvantages.

Problem 3.36 (Greedy vs. Hill Climbing)
What is the fundamental difference between Greedy Search and Hill Climbing? Explain.

Problem 3.37 (Local Beam Search)
What known algorithm does Local Beam Search become if $k = 1$?

Problem 3.38 (Hill Climbing) 40pt
Consider a world with equal number of women and men. Every man is interested in a nonnegative number of women and vice versa. You are given a matrix that specifies a directed graph of *interest* between the people. Write an SML function that uses local search to find a pairing `{<man,woman>,<man,woman>,...}` such that no man is paired up with >1 women and vice versa. A pairing is admissible if in every pair `<man i, woman j>` the two people are interested in each other. An optimal pairing is the pairing with the highest cardinality of all the possible pairings in a problem.

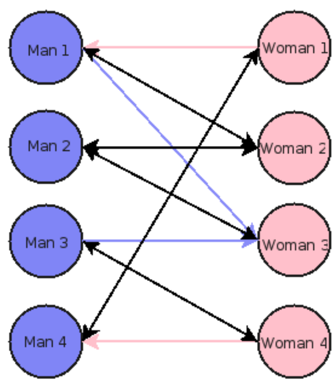
To accomplish this task follow the steps outlined below:

- Define what is a state in this problem
- Given any state, describe what the neighbours of this state are (i.e. describe how neighbours are related). Hint: think about neighbours in the Traveling Salesman Problem
- Find and describe a heuristic. What is the optimal value of your heuristic?

- Write an SML function pairup that takes an interest graph (represented as a matrix) and an initial paring (not necessarily admissible) and uses hill climbing to return an admissible pairing. A sample hill-climbing algorithm is provided in the slides. You may assume that the format of the input matrix is correct

Input: The following matrix encodes the graph below:

	Woman			
Man	$< 0,1 >$	$< 1,1 >$	$< 1,0 >$	$< 0,0 >$
	$< 0,0 >$	$< 1,1 >$	$< 1,1 >$	$< 0,0 >$
	$< 0,0 >$	$< 0,0 >$	$< 1,0 >$	$< 1,0 >$
	$< 1,1 >$	$< 0,0 >$	$< 0,0 >$	$< 0,1 >$



The first value indicates if the man is interested in the woman, while the second value indicates if the woman is interested in the man.

It would be encoded as follows:

```
val matrix = [[(false,true),(true,true),(true,false),(false,false)],[(false,false),(true,true),(true
[(false,false),(false,false),(true,false),(true,true)],[(true,true),(false,false),(false,false),(fal
```

Use the following datatypes:

```
datatype man = man of int
datatype woman = woman of int
type pairing = (man * woman) list
type matrix = (bool * bool) list list
```

Function signature:

```
val pairup = fn : pairing -> matrix -> pairing
```

Sample run:

```
val matrix = [[(false,true),(true,true),(true,false),(false,false)],[(false,false),(true,true),(true
[(false,false),(false,false),(true,false),(true,true)],[(true,true),(false,false),(false,false),(fal
val init = [(man 1,woman 2),(man 2,woman 3),(man 3,woman 1),(man 4,woman 4)];
pairup init matrix;
(*Ideally*)
val it = [(man 1,woman 2),(man 2,woman 3),(man 3,woman 4),(man 4,woman 1)] : (man * woman) list
```

Problem 3.39 (Easter Bunnies in Boxes)

Imagine there are n Easter bunnies and n different coloured boxes, and each bunny has specific color preferences and will like their box on a scale of 1 to 10. We want to makes as many bunnies as happy as we can, so the overall fitness of an assignment of bunnies in boxes will be the sum of how much each bunny likes its box. An assignment is admissible if each bunny has exactly 1 box. Think about applying Genetic Algorithms for this problem: your task is to come up with an encoding that allows only admissible states and with crossover and mutation operators that preserve admissibility. Don't take the term crossover too literally though - it is not a must that you split the chromosomes and cross over their parts, you can think about the concept of reproduction in general. Similarly for mutation.

Problem 3.40 (Implementing simulated annealing)
Write an SML function that implements the simulated annealing algorithm to find the x value where a function $f(x)$ has a maximum. Your function should take the following arguments:

- `f : real->real` the SML implementation of $f(x)$
- `(a,b) : real*real` an interval $[a;b]$ in which to search for the maximum
- `schedule : int->real` a function that maps time steps to temperature values

For example the maximum of $f(x) = -(x-2)^2$ in $[0.0;5.0]$ is at $x = 2.0$. Given a good temperature schedule your implementation should be able to compute the maximum of $sin(x)$ with an accuracy of 0.0001. Show this at the end of your program by computing the maximum of $sin(x)$ in the interval $[0.0;5.0]$.

The complete signature of the function should look like this:

```
find_max : (real -> real) -> real * real -> (int -> real) -> real
```

Problem 3.41 (Simulated annealing schedules)
In the simulated annealing algorithm one has to choose a temperature schedule. Two possible schedules are:

- **The linear cooling scheme:** $T_{k+1} = T_k - \alpha = T_0 - (k+1) * \alpha$
- **The exponential cooling scheme:** $T_{k+1} = \alpha T_k = \alpha^{k+1} T_0$ where $\alpha < 1.0$ (the typical value is 0.95, but this really depends on the problem - and the smaller this is, the less iterations you will have).

The exponential cooling scheme typically performs better. Explain why this might be the case. To help you with this you should do an experiment where you try to achieve the desired accuracy in the pevious question by using both a linear and an exponential schedule.

Problem 3.42 (Simulated Annealing)
Assume that you are using Simulated Annealing to solve the 8Queens problem. The SA is at a point where $T = 3$, the energy (fitness) of the current state is $E_{current} = 7$ and the energy of the neighboring state is $E_{neighbor} = 4$. With what probability will the neighbor be accepted as the new state and why?

3.2 Logic Programming

3.2.1 Introduction to Logic Programming and PROLOG

nothing here

3.2.2 Programming as Search

These exercises should be tried by everybody. They will confront you with the main (conceptual) problems of programming PROLOG, like relational programming, recursion, and a term language.

Problem 3.43: Build a database of facts about flight connections from Bremen Airport and write some query predicates for connections. Consider it is furthermore plausible to assume that whenever it is possible to take a flight from A to B, it is also possible to take a flight from B to A.

3.2.3 Logic Programming as Resolution Theorem Proving

No problems supplied yet.

References

[Gen11a] General Computer Science; 320101: GenCS I Lecture Notes. Online course notes at http://kwarc.info/teaching/GenCS1/notes.pdf, 2011.

[Gen11b] General Computer Science; Problems and Solutions for 320101 GenCS I. Online practice problems with solutions at http://kwarc.info/teaching/GenCS1/solutions.pdf, 2011.

[Gen11c] General Computer Science: 320201 GenCS II Lecture Notes. Online course notes at http://kwarc.info/teaching/GenCS2/notes.pdf, 2011.

[Gen11d] General Computer Science: 320201 GenCS II Lecture Notes. Online practice problems with solutions at http://kwarc.info/teaching/GenCS2/solutions.pdf, 2011.

[Gen11e] General Computer Science: 320201 GenCS II Lecture Notes, 2011.