General Computer Science 320201 GenCS I & II Problems

Michael Kohlhase

Computer Science
Jacobs University, Bremen Germany
m.kohlhase@jacobs-university.de

September 21, 2016

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 2003-2016. It is meant as a supplement to the course notes [Koh11a, Koh11b]. 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 [Koh11e, Koh11c]

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 Cîrlănaru, 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

Chapter 1

Getting Started with “General Computer Science”

1.1 Overview over the Course

This should pose no problems

1.2 Administrativa

Neither should the administrativa

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.

Chapter 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.

Part I

Representation and Computation

Chapter 3

Elementary Discrete Math

3.1 Mathematical Foundations: Natural Numbers

Problem 0.7 (A wrong induction proof)
What is wrong with the following "proof by induction"? 25pt

> **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 $\square$

Problem 0.8 (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 0.9 (Peano's induction axiom)
State Peano's induction axiom and discuss what it can be used for.

3.2 Reasoning about Natural Numbers

Problem 0.10 (Zero is not one)
Prove or refute that $s(o)$ is different from o.

Note: Please use **only** the Peano Axioms for this proof.

Problem 0.11 (Natural numbers)
Prove or refute that $s(s(o))$ and $s(s(s(o)))$ are unary natural numbers and that their successors are different.

3.3 Defining Operations on Natural Numbers

Problem 0.12 Figure out the functions on natural numbers for the following defining equations

$$\tau(o) = o$$

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

Problem 0.13 (Commutativity of addition)
Prove by induction or refute the commutativity of addition in the case of natural numbers using only its definition from the slides. More specifically prove that: $n \oplus m = m \oplus n$

Hint: You might come to a point in the proof where a new subproblem emerges which needs a to be proven by induction. Better said, you are allowed (and suppose to) use **nested** induction.

Problem 0.14 (Unary Natural Numbers)
Let $\oplus$ be the addition operation and $\odot$ be the multiplication operation on unary natural numbers as defined on the slides. Prove or refute that:

1. $a \oplus b = b \oplus a$
2. $(a \oplus b) \odot c = a \odot c \oplus b \odot c$
3. $a \odot b = b \odot a$

3.4 Naive Set Theory

25pt **Problem 0.15** 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 0.16** 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.

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

20pt **Problem 0.17** 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 $y \in (B \cap A)$, $(x, y) \in P \Leftrightarrow (x, y) \in R \vee (x, y) \in Q$. Prove or refute (by giving a counterexample) the following statement: If Q and P are total functions, then P is a partial function.

3.5 Talking (and writing) about Mathematics

Problem 0.18 Fill in the blanks in the table of Greek letters. Note that capitalized names denote capital Greek letters. 3pt

3min

Symbol					γ	Σ	π	Φ
Name	alpha	eta	lambda	iota				

Problem 0.19 (Math Talk)
Write the following statement in mathtalk

"For all mandatory courses, there is a student such that if the student is from the major in which the course is mandatory then the student is not taking that course."

3.6 Relations

Problem 0.20 (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

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

Problem 0.21 (Meet the Transitive Closure)
The transitive closure R^* of a binary relation R on a set S is the smallest transitive relation on S that contains R.

Prove or disprove that given two equivalence relations R_1 and R_2 on the set S:

a. $R_1 \cup R_2$ is an equivalence relation.

b. $R_1 \cup {R_2}^*$ is an equivalence relation.

Problem 0.22 (Relation Properties)
Given a base set $A := \{a, b, c\}$, for each of the following properties:

1. reflexive
2. symmetric
3. antisymmetric
4. transitive

write down a relation $R_i \subseteq A \times A$ $(i = 1, \ldots, 4)$ that has property i and contains at least three tuples.

3.7 Functions

Problem 0.23 Are the following functions total, injective, surjective and/or bijective?

- $f:\mathbb{R}\to\mathbb{R}, f(x)=x^2-3x+6$
- $g:\mathbb{N}\to\mathbb{N}, g(x)$ represents the number of distinct prime divisors of x
- $h\colon \mathbb{N}^+\times\mathbb{N}^+\to\mathbb{N}^+$,

$$h(m,n):=\frac{(m+n-2)(m+n-1)}{2}+m$$

Prove your answers.

Problem 0.24 (Function Property)
Prove or refute: if a mapping of a finite set to itself is injective then it is surjective as well. 15pt

Problem 0.25 (Composition of functions and relations)
Prove or refute that the composition of functions, as defined in the lecture, is compatible with the definition of functions as special relations. That is: Let $F\subseteq A\times B$ and $G\subseteq B\times C$ be relations that are total functions, and let $H\subseteq A\times C$ be defined as $H:=G\circ F$, i. e. the composition of the relations F and G. Show that

1. the relation H is a total function.
2. for all $x\in A$, we have $h(x)=g(f(x))$.

Note: Note: F, G and H are written in capital letters here to point out that we are talking about relations. If we treat them as functions, they would usually be written in lowercase.

Chapter 4

Computing with Functions over Inductively Defined Sets

4.1 Standard ML: Functions as First-Class Objects

Problem 0.26 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 0.27 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 ?prob.member?

Problem 0.28 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] ⇝ [[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]] ⇝ [1,2,3] [4,5,6]. 20pt

Problem 0.29 (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 0.30 (Decompressing binary lists)
Write an inverse function UNZIPit of the one written in ?prob.zipbin?.

Problem 0.31 Program the function f with $f(x) = x^2$ on unary natural numbers without using the multiplication function. 15pt

Problem 0.32 (Translating between Integers and Strings)
SML has pre-defined types int and string, write two conversion functions: 20pt

- int2string converts an integer to a string, i.e. int2string(˜317) ⇝ "˜317":string
- string2int converts a suitable string to an integer, i.e. string2int("444") ⇝ 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 0.33 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 0.34 Write a non-recursive variant of the member function from ?prob.member? using the foldl function.

Problem 0.35 (Decimal representations as lists)

15pt

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.

Problem 0.36 (List functions via folding)

30pt

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 the previous problem.

Problem 0.37 (Mapping and Appending)

10pt

Can the functions mapcan and mapcan2 be written using foldl/foldr?

4.2 Inductively Defined Sets and Computation

Problem 0.38 Figure out the functions on natural numbers for the following defining equations

$$\tau(o) = o$$

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

Problem 0.39 (A function on natural numbers)
Figure out the operation η on unary natural numbers defined by the following equations: 15pt

5min

$$\eta(o) = o$$

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

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

Problem 0.40 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. 15pt

4.3 Inductively Defined Sets in SML

8pt 8min **Problem 0.41** 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.

Use SML syntax for the whole problem.

4pt 8min **Problem 0.42** 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 ms, i.e. nms(s(s(zero)),s(zero)) evaluates to construct(s(zero),construct(s(zero),elist)).

20pt **Problem 0.43** Given the following SML data type for an arithmetic expressions

```
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 0.44 (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 0.45 (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 0.46 (Nary Multiplication)
By defining a new datatype for n-tuples of unary natural numbers, implement an n-ary multiplications using the function mul from ?prob.natoper?. 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 2 2 3 = 12.

Chapter 5

Abstract Data Types and Term Languages

5.1 Abstract Data Types and Ground Constructor Terms

Problem 0.47 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 0.48 Translate the given SML datatype 5pt

```
datatype T = 0 | c1 of T * T | c2 of T -> (T * T)
```
5min

into abstract data type in mathmatical notation.

Problem 0.49 (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 $[3,4,[7,[8,2],9],122,[2,2]]$. 20pt

5.2 A First Abstract Interpreter

30pt **Problem 0.50** Give the defining equations for the maximum function for two numbers. This function takes two arguments and returns the larger one.

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

15pt **Problem 0.51** Using the abstract data type of truth functions from ?prob.truth-values?, 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.

6pt **Problem 0.52** Consider the following abstract data type:

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

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
$f(g(a))$			
$f(g(\langle a, b\rangle))$			
$h(g(\langle h(x_{\mathbb{C}}), f(c)\rangle))$			
$h(g(\langle h(x_{\mathbb{B}}), f(y_{\mathbb{C}})\rangle))$			

5.3 Substitutions

Problem 0.53 (Substitution)
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 $\tau(t)$). 4pt 5min

Definition 5.3.1 We call a substitution σ **idempotent**, iff $\sigma(\sigma(\mathbf{A})) = \sigma(\mathbf{A})$ for all terms $\mathbf{A}$.

Definition 5.3.2 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 $\mathbf{supp}(\sigma) := \{x_i \mid 1 \leq i \leq n\}$

Problem 0.54 Prove or refute that σ is idempotent, if $\mathbf{intro}(\sigma) \cap \mathbf{supp}(\sigma) = \emptyset$. 30pt

Problem 0.55 (Substitution Application)
Consider the following SML data type of terms: 30pt

```
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 $[y/x], [x/z](f(z)) = f(x)$.

5.4 Terms in Abstract Data Types

5.5 A Second Abstract Interpreter

Problem 0.56 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 $\mathcal{P}$?

5.6 Evaluation Order and Termination

4pt 10min **Problem 0.57** 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.

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

2pt 5min **Problem 0.58** 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.

15pt **Problem 0.59** Give the recursion relation of the abstract procedures in ?prob.square?, ?prob.truth-values?, ?prob.if?, and ?prob.max? and discuss termination.

Chapter 6

More SML

6.1 More SML: Recursion in the Real World

No problems supplied yet.

6.2 Programming with Effects: Imperative Features in SML

6.2.1 Input and Output

nothing here yet.

6.2.2 Even more SML: Exceptions and State in SML

Problem 0.60 (Integer Intervals)
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. 5pt 10min

Problem 0.61 (Operations with Exceptions)
Add to the functions from ?prob.natoper? 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 0.62 (List Functions with Exceptions)
Write three SML functions nth, take, drop that take a list and an integer as arguments, such that 6pt 20min

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 0.63 (Transformations with Errors)
Extend the function from ?prob.ML-int2string? 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 0.64 (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 10pt

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 0.65 (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 and Real.fromString but do the string parsing yourself. You may however use the char_to_int from above.

- 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 0.66 (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
```

Problem 0.67 (List evaluation)

10pt 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]
```

Problem 0.68 (String parsing)

10pt 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]
```

Problem 0.69 (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. 10pt

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
```

Bibliography

[Koh11a] Michael Kohlhase. General Computer Science; 320101: GenCS I Lecture Notes. Online course notes at http://kwarc.info/teaching/GenCS1/notes.pdf, 2011.

[Koh11b] Michael Kohlhase. General Computer Science: 320201 GenCS II Lecture Notes. Online course notes at http://kwarc.info/teaching/GenCS2/notes.pdf, 2011.

[Koh11c] Michael Kohlhase. General Computer Science: 320201 GenCS II Lecture Notes. Online practice problems with solutions at http://kwarc.info/teaching/GenCS2/solutions.pdf, 2011.

[Koh11d] Michael Kohlhase. General Computer Science: 320201 GenCS II Lecture Notes, 2011.

[Koh11e] Michael Kohlhase. General Computer Science; Problems and Solutions for 320101 GenCS I. Online practice problems with solutions at http://kwarc.info/teaching/GenCS1/solutions.pdf, 2011.