

# General Computer Science II Retake Spring 2016

Michael Kohlhase  
Jacobs University Bremen  
FOR COURSE PURPOSES ONLY

February 20, 2016

## Contents

1	Assignment 1 (Graph Theory and Circuits) –	2
2	Assignment 2 (Combinatorial Circuits and PNS) –	3
3	Assignment 3 (Combinatorial Circuits and PNS) –	4
4	Assignment 4 (ASM Language) –	6
5	Assignment 5 (ASM Language) –	7
6	Assignment 6 (L(VMP) and $\mu$ ML) –	9
7	Assignment 7 (Turing Machines) –	13
8	Assignment 8 (Internet and Mail Services) –	15
9	Assignment 9 (WWW) –	17
10	Assignment 10 (Encryption and Search) –	18
11	Assignment 11 (Astar-search) –	22

# 1 Assignment 1 (Graph Theory and Circuits) –

## Problem 1.1 (In-degrees in acyclic digraphs)

Prove by induction or refute that any acyclic digraph with non-empty set of nodes has at least one node with in-degree 0. 30pt

## Problem 1.2 (Depth of a Fully Balanced Binary Tree)

Prove or refute that in a fully balanced binary tree with  $n \geq 1$  nodes, the depth is  $\lceil \log(n) \rceil$ . 15pt

## Problem 1.3 (Generate Boolean Tree)

Write an SML function `MakeBTree` which takes an odd integer  $n$  and returns a balanced binary tree with  $n$  nodes of the type `datatype btree = leaf | parent of btree*btree` 25pt

**Problem 1.4** Using Quine McCluskey and shared monomials make a circuit for calculating the values of  $f_1$  and  $f_2$ , given by the following table, for input  $(x_1, x_2, x_3)$ . 30pt

$x_1$	$x_2$	$x_3$	$f_1$	$f_2$
0	0	0	0	1
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	1	1
1	0	1	0	0
1	1	0	1	1
1	1	1	0	0

## 2 Assignment 2 (Combinatorial Circuits and PNS) –

### Problem 2.1 (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`

**Problem 2.2** Design a two bit multiplier using only NOT, OR, AND and XOR gates. 10pt

### Problem 2.3 (6-ary AND gate)

Using only normal AND gates draw the inside of a 6-ary AND gate. What is the minimal depth of such a circuit and why? 5pt

### Problem 2.4 (Conditional circuit)

Design a 4-bit conditional circuit that implements the following operation: 20pt

if  $x \leq y$  then  $x + y$  else  $2x$

### Problem 2.5 (Periodic Number Representation)

Prove that if the base- $b$  representation of a real number  $x = x_0.x_1x_2x_3\dots_b$  is periodic then  $x$  is a rational number. The representation is periodic if there are positive integers  $n$  and  $p$  such that  $x_k = x_{k+p}$  for all integers  $k > n$ . For example,  $10.11_2 = 2\frac{3}{4}$ ,  $0.112_3 = 0.112112112\dots_3 = \frac{7}{13}$ , and  $2.012_8 = 2.01222222\dots_8 = 2\frac{9}{448}$ . 35pt

### 3 Assignment 3 (Combinatorial Circuits and PNS) –

#### Problem 3.1 (Incremental Adder)

25pt

Design a circuit that recursively adds consecutive numbers, i.e. it first adds 1, then 2, then 3 etc. The starting number is zero and all the numbers are represented with 4 bits. There is no need to deal with overflows.

The outputs should be 1, 3, 6, 10, 15, 5 ... etc.

#### Problem 3.2 (Detecting overflow in TCN addition)

10pt

1. Convert the following pairs of decimal numbers into (4-bit) two's complement notation and add them (to obtain a 4-bit result). Convert the sum back to decimal and check whether the answer you obtained is correct. Thus state whether the addition was proper or an overflow/underflow occurred.
  - -1 and 7
  - -5 and -4
2. According to the TCN Main Theorem, when adding two  $n$ -bit TCN numbers, a proper sum is obtained when the last two carry bits are equal. Otherwise an overflow or an underflow occurred. There are a number of other ways to detect that, however. Consider the full adder used for the addition of the most significant bits of the two numbers. Look at the inputs and outputs of this full adder, and determine when an overflow occurs depending on the carry-in and carry-out bit values. Write down the logic equation for this and draw its corresponding combinatorial circuit.

#### Problem 3.3 (TCN in SML)

Given the datatype: `type tcn = int list` write the following SML functions

20pt

- `extend = fn : tcn -> int -> tcn` which takes a number in Two's complement representation and an integer  $n$  and makes the tcn number  $n$  bits wide. Here  $n$  should always be bigger or equal to the current width of the number.
- `int2tcn = fn : int -> int -> tcn` which converts an integer number to a tcn number given the width.
- `tcn2int = fn : tcn -> int` which converts a tcn number to an integer number.

If in any of the functions a number can't fit into the requested number of bits raise the `NumberDoesNotFit` exception.

---

**Note:** A number's width is simply the number of bits that are used to represent that number.

---

As an example consider

```
int2tcn 10 8 -> [0,0,0,0,1,0,1,0]
```

#### Problem 3.4 (Accumulator Circuit)

To finish off your work on designing circuits, here is an all-encompassing assignment. You have to design an accumulator. 35pt

It should be implemented as a circuit which takes an 8-bit input value, together with a 2-bit control input. When this control input is "off" (i.e. its value is 00), the accumulator should store the 8-bit input value. When the control input is 10, the accumulator should

add the 8-bit input value to the value previously stored in it, and save the new result. Similarly, when the control input is 11, the accumulator should subtract the 8-bit value from its old value. You are supposed to use a CSA for this problem.

---

**Note:** Since this is a problem that is supposed to summarize all your knowledge on circuits from GenCS, please create your circuit out of gates. You will need building blocks like flip-flops and adders, but please draw their detailed circuits (you can copy from the slides), and then use some short notation for them.

---

## 4 Assignment 4 (ASM Language) –

**Problem 4.1** Write an ASM program that multiplies numbers at addresses 0 and 1 and stores the product at address 2. Show intermediate states of registers/memory. 10pt

**Problem 4.2 (Fibonacci Numbers)**

Write an ASM program that computes the  $n^{\text{th}}$  Fibonacci number in the accumulator, where  $n$  is the value in cell 0. You do not have to worry about sizes of memory cells here. Just assume that there is enough space. 20pt

**Problem 4.3 (Longest equal sublist)**

Write an ASM program which determines the length of the longest sequence of consecutive equal numbers from a given list of  $n$  numbers and the number itself. 30pt

For example, the answer for this list: 1, 1, 2, 5, 2, 2, 2, 6, 6 is length 3 and number 2.

D(0) contains the number of elements in the list. It is guaranteed that the list contains at least one element. D(10) through D(10 +  $n$  - 1) contain the elements of the list. At the end of the execution, the length must be saved in D(1) and the number in D(2)

Note that you are only allowed to use labels in jump statements. You are very much encouraged to write comments and even a description of your approach, which can help the grader award partial points.

load i	jump(=) i
store i	jump(>) i
loadi i	jump(<) i
addi i	jump( $\geq$ ) i
subi i	jump( $\leq$ ) i
add i	sub i
move S T	jump( $\neq$ ) i
loadin i j	storein i j
jump i	stop 0

You can find a list of ASM commands (only for reference) on the right.

**Problem 4.4 (Binary to decimal)**

Let D(0) =  $n$  contain the number of bits of a binary number stored in D(2) . . . D(2 +  $n$  - 1). Each memory cell represents one bit of the number where D(2) is the least significant bit and D(2 +  $n$  - 1) is the most significant bit. Write a program that stores the corresponding decimal number in D(1). 40pt

## 5 Assignment 5 (ASM Language) –

### Problem 5.1 (Convert Highlevel Code to $\mathcal{L}(\text{VM})$ Code)

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

20pt

```
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  $\mathcal{L}(\text{VM})$  code.

### Problem 5.2 (Simulating REMA in SML)

40pt

Given the following declarations:

```
datatype register = acc | in1 | in2;
datatype instr = load of int | loadi of int | loadin1 of int | loadin2 of int |
               store of int | storein1 of int | storein2 of int |
               add of int | addi of int | sub of int | subi of int |
               move of register*register | nop of int | stop of int |
               jump of int | jumpe of int | jumpne of int |
               jumpl of int | jumple of int | jumpg of int | jumpge of int;
type program = instr list;
type memory = int list;
```

```
(* This is the state of the machine. From left to right the values mean:
   PC register; ACC register; IN1 register; IN2 resigter; Memory cells*)
type state = int*int*int*int*(int list);
```

Write two SML functions:

- `execute_instr : instr -> state -> state`
- `run : program -> memory -> memory`

The first function takes an ASM instruction and the current state of the REMA as arguments and returns the new state after the instruction is executed. The second function takes a program and the initial configuration of the memory. It then simulates the program until a `STOP 0` instruction is reached and returns the memory at that point. In both functions 'memory' is just a list of integers that represent the current state of the memory of the REMA. Once the initial list is supplied, during simulation its length shouldn't change.

**Note:** For this problem and the next it will be very helpful to use built-in SML functions. Make sure to check the forums for more info.

### Problem 5.3 (SW TCN Converter)

Write a Simple While program that converts TCN to decimal. You will be given the TCN number  $t$  (consider it a variable set at the beginning of your program). You can consider that the number  $t$  is an integer formed only from 1s and 0s and the most significant bit is represented by the first digit; remember that the first bit also tells you whether the number is negative or positive.

25pt

Your program should start, for example, with:

```
var t=11001;
```

That is,  $t = 11001$ , which corresponds to  $-7$ .

**Problem 5.4 (Simple While program on Fibonacci)**

15pt

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

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



## 6 Assignment 6 (L(VMP) and $\mu$ ML) –

### Problem 6.1 (Simple While with procedures)

Let SWP (Simple While with procedures) be an imperative language that supports function calls. Each of the functions/procedures is defined like a standard SW program: a list of variable declarations, a statement or sequence of statements and a return value. The only difference is that you now also have a list of arguments. Here is a program, written in concrete syntax that has one procedure which computes the factorial function and the main program uses it to calculate  $10! + 1$ . 40pt

```
fun fact(n)
  var i:=1;
  var f:=1;
  while i<=n do
    f:=f*i;
    i:=i+1;
  end
  return f;

main
  var r:=0;
  r:=fact(10)+1;
  return r;
```

Let the abstract syntax of this language be defined with the following types/datatypes:

```
type id = string (* identifier *)
```

```
datatype exp = (* expression *)
  Con of int (* constant *)
| Id of id (* variable or argument *)
| Add of exp * exp (* addition *)
| Sub of exp * exp (* subtraction *)
| Mul of exp * exp (* multiplication *)
| Leq of exp * exp (* less or equal test *)
| App of id * exp list (* application *)
```

```
datatype sta = (* statement *)
  Assign of id * exp (* assignment *)
| If of exp * sta * sta (* conditional *)
| While of exp * sta (* while loop *)
| Seq of sta list (* sequentialization *)
```

```
type declaration = id * exp (* variable declaration *)
```

```
type procedure = id * id list * declaration list * sta * exp (* procedure declaration *)
```

```
type programBody = declaration list * sta * exp (* main procedure *)
```

```
type program = procedure list * programBody
```

The same example written in abstract syntax:

```

val factprog = ([("fact",["n"],[ ("i", Con(1)), ("f", Con(1)) ] ,
                While(Leq(Id("i"), Id("n")),
                      Seq([Assign("f", Mul(Id("f"), Id("i"))),
                          Assign("i", Add(Id("i"), Con(1))) ])),
                Id("f")) ],
  ( [("r", Con(0))],
    Assign("r", Add( App("fact", [Con (10)]), Con(1) )),
    Id("r") ) )

```

This language can be compiled using a combination of  $\mathcal{L}(\text{VM})$  and  $\mathcal{L}(\text{VMP})$ . The problem with `peek` and `poke` inside procedures can be solved in the same way SW is compiled - first make the list of initially declared values and then use `peek` for variable reference, `poke` for variable assignment and `arg` for argument reference. The arguments of `peek` and `poke` will now be referencing indexes in the stack with respect to the current frame, i.e. from 1 to  $n$ , where  $n$  is the number of variable declarations. The only exception is with the main program where there is no return address stored in the current frame, so the numbering will start from 0. `return` will delete the variables along with the arguments and return address.

Your task is to write an SML function `compileSWP` which takes a SWP program in abstract syntax as an argument and compiles it into  $\mathcal{L}(\text{VM}) + \mathcal{L}(\text{VMP})$  (with the abstract syntax defined on the slides).

Signature and example:

```

val compileSWP = fn : program -> code

- compileSWP(factprog);
val it =
  [proc (1,33),con 1,con 1,arg 1,peek 1,leq,cjp 18,peek 1,peek 2,mul,poke 2,
   con 1,peek 1,add,poke 1,jp ~21,peek 2,return,con 0,con 1,con 10,call 0,add,
   poke 0,peek 0,halt] : code

```

### Problem 6.2 ( $\mu$ ML Mystery)

You are given the following piece of  $\mu$ ML code:

30pt

```

(
  [
    ("helper1", ["x", "y"],
      If(Leq(Id"y", Con 0), Con 0,
        If(Leq(Id"x", Id"y"), App("helper1", [Id"x", Sub(Id"y", Id"x")]), Con 1))),
    ("helper2", ["x", "n"],
      If(Leq(Id"n", Con 2), App("helper1", [Con 2, Id"x"]),
        If(App("helper1", [Id"n", Id"x"]),
          If(App("helper2", [Id"x", Sub(Id"n", Con 1)]), Con 1, Con 0),
          Con 0)
        )
    ]
  ),
  ("myproc", ["x"],
    App("helper2", [Id"x", Sub(Id"x", Con 1)])
  )
)

```

```

)
],
App("myproc", [Con 6])
);

```

1. State what the program  $myproc(x)$  is doing. (Assume  $x > 1$ .)
2. Write another  $\mu$ ML program that does the same thing in a different way. Remember that a  $\mu$ ML program should be a pair of a well defined list of function declarations, and a single `App` call to the main function. Of course, that function will be calling the helping function(s) in its body, and the helping functions may call themselves.

**Problem 6.3 (SML simulator for  $\mathcal{L}(\text{VM})$  with static procedures)**

Write an SML function `vm_simulator` that simulates a program written in Virtual Machine Language. It should take as input:

- a string list, a list of  $\mathcal{L}(\text{VM})$  commands including `con  $i$` , `add`, `sub`, `mul`, `leq`, `jp  $i$` , `cjp  $i$` , `peek  $i$` , `poke  $i$` , `proc  $i$   $j$` , `arg  $i$` , `call  $i$` , `return`, `halt` and numbers. Note that, for example, `"con "` and `"5"` count as 2 instructions.
- an int list describing the initial state of the stack; the first element in the list represents the top of the stack while the last element represents bottom of the stack. Of course it can also be initially empty.

```
val vm_simulator = fn : string list * int list -> int list
```

As there are a few things that could possibly go wrong, make sure to use the following exceptions:

- `NumberExpected`, if no number follows after `con`, `jp`, `cjp`, `peek`, `poke`.
- `InvalidPeekPoke`, if the number following `peek` or `poke` is smaller than 0 or bigger than the size of the stack.
- `InvalidJump`, if the number following `jp`, `cjp`, `call` or `proc` makes the virtual program counter point to an unexisting instruction.
- `InvalidStackSize`, if the stack is not big enough so that the current instruction can be normally executed. (`add`, `sub`, `leq` etc.)
- `UnknownCommand`, if the input string list contains strings that are not one of the VML commands described in class.

Signature and examples:

```
val vm_simulator = fn : string list * int list -> int list
```

```
vm_simulator(["con", "~2", "con", "~7", "mul", "con", "3", "con", "4", "mul", "sub"], [2, 9, 7]);
val it = [~2, 2, 9, 7] : int list
```

```
vm_simulator(["proc", "2", "9", "arg", "1", "arg", "2", "add", "return",
              "con", "1", "con", "2", "con", "3", "con", "4",
              "call", "0", "call", "0", "call", "0", "halt"], []);
val it = [10] : int list
```

**Note:** You can safely assume that in each test case only 1 version of  $\mathcal{L}(\text{VM})$  will be used, i.e. either  $\mathcal{L}(\text{VM})$  with `peek  $i$`  and `poke  $i$`  or  $\mathcal{L}(\text{VM})$  with `proc  $i$   $j$`  and `call  $i$` .



## 7 Assignment 7 (Turing Machines) –

### Problem 7.1 (Divisible by three in Turing Machine)

Design a Turing Machine that determines whether a number is divisible by 3. Your TM is 30pt  
 over alphabet  $0, 1, \#, Y, N$ , and you can assume that your head is initially positioned on  
 the first digit of a binary representation of a non-negative number which is written between  
 two hash symbols ( $\#$ ). Upon termination, your program should replace the second hash  
 with  $Y$  or  $N$  indicating that the number whose binary representation is written on tape is  
 divisible by three or not. So if the tape was

$$\# \overbrace{1}^{Hd} 010\#$$

then after execution we should have:

$$\#1010 \overbrace{Y}^{Hd}$$

It is mandatory that you explain your approach in plain English.

### Problem 7.2 (Palindrome Detector Turing Machine)

Create a Turing machine that detects whether the word  $w \in \{0, 1\}^*$  on the tape is a 25pt  
 palindrome, i.e.  $w = w^R$ , where  $w^R$  is  $w$  reversed. If so, leave a 1 at the final position of  
 the head, which can be anywhere on the tape; otherwise leave a 0. Assume that, initially,  
 the head is over the first character of  $w$ , and that  $w$  is surrounded by hash marks as  
 delimiters, i.e. you have the alphabet  $\{0, 1, \#\}$ .

---

**Note:** Admissible moves are  $L$  (*left*),  $R$  (*right*), and  $N$  (*none*) with the obvious meaning.

---

### Problem 7.3 (Various Bracket structure)

Design the following Turing Machine. Its input is a word  $w \in \{(\,), [\,], \#\}^*$ , surrounded by 30pt  
 hash marks as delimiters. You can assume that your head is initially positioned right after  
 the first hash. Upon termination, your program should replace the second hash with  $Y$  or  
 $N$  indicating if the bracket structure is correct ( $Y$  - for correct,  $N$  otherwise). Thus the  
 overall alphabet is  $\{(\,), [\,], \#, Y, N\}$ .

---

**Note:** Correct bracket structure is a sequence of the same amount of opening and closing  
 brackets, where the amount of opening brackets is not less than the amount of closing brackets  
 in any prefix of such a sequence and they are positioned in the proper order.

Example:

initial	#[[()]#
result	# - something - N
initial	#[()[]#
result	# - something - Y

---

### Problem 7.4 (Counting Characters)

Given a string of a's and b's delimited by  $\#$  (e.g.  $\#_a b a a a b b a b b a \#$ ) with the head (un- 15pt

derlined) initially over the leftmost letter, write a Turing machine program that finds out whether there are more a's than b's or vice versa. Return the result as one character under the head when the machine stops: An "a" for "more a's", a "b" for "more b's", or a "#"  
for an equal number.

## 8 Assignment 8 (Internet and Mail Services) –

### Problem 8.1 (SMTP Mail Writer)

You have recently learned in the lecture about how you can connect to a SMTP server via Telnet and send a simple email message. Your first task is now to automate the process by creating an SML function `sendMail` that will: 35pt

1. open a connection (socket) to the server
2. issue a 'HELO' command for self-identification
3. start a mail message using 'MAIL FROM'
4. set the recipient ('RCPT') and mail contents ('DATA')
5. close the connection using 'QUIT'

Remember that the server will understand a single point on a line as a mark of mail ending!

Your function should have the following signature:

```
val sendMail = fn : string * int * string list * string -> bool
```

The parameters, in order, are: the hostname, the port number (usually 25), a list of mail recipients, and the message body. The function should return true if everything was OK or false if there were problems. You are not required to treat any exceptions raised by SML library functions in your implementation.

### Problem 8.2 (POP Mail Reader)

Write the following SML functions:

35pt

1. `countMails`, which returns the number of emails available on the server. Consider that you have to first login using your username and password, so you will first to send the username and password. The function must have the following signature:

```
val countMails = fn : string * int * string * string -> int
```

The arguments are, in order: the string identifying the host, the port number (usually 110), the user name and the password. The return value is the number of emails available on the server.

2. `readMail`, which returns the contents of one of the emails available on the server. The function must have the following signature:

```
val readMail = fn : string * int * string * string * int -> string
```

The arguments are, in order: the string identifying the host, the port number, the user name and password and the number of the email that you are going to request.

The return value is the content of the email you requested.

The email contains a set of meta-data containing, for example, the sender, the subject, and date. You can simply leave them in the output string for now.

### Problem 8.3 (Email relay server)

Now that you have implemented mail sending and retrieving, consider writing an email relay server. Theoretically, it should be contacted by an email client in order to implement routing of the messages between the sending client to the destination server. 35pt

In your case, you have to write an SML function `routeMails` that will accept incoming connections and requests of email sending. Your function will emulate an SMTP server for a limited set of commands (the ones that you implemented in the SMTP client are enough). After getting the contents of the email, it will simply re-transmit it further to the destination email address (that you have to read from the message you receive).

The function will have the following signature:

```
val routeMails = fn : int * string -> bool;
```

The `int` represents the port number the server will open to, while the `string` represents the address of the server the email is going to be forwarded to. The function return value should be `true` in case everything went OK, and `false` in case of error. Alternatively, you can leave thrown exceptions uncaught to signal an error. Here is an example run:

```
(* you run this function first *)
routeMails(2525, 'exchange.jacobs-university.de');

(* then, in another terminal, you run this *)
sendMail('localhost', 2525, ['youraddress@jacobs-university.de'],
        'Test');
```

You should wait for a short period of time and then check your Jacobs mail. You should see the email there!

---

**Note:** The server should wait in a loop for the next message to come - that means you should not finish execution after the first email has been forwarded.

---



## 9 Assignment 9 (WWW) –

### Problem 9.1 (Final practice)

40pt

The final exam is not more than a month away so it is good to start practicing. To do this you decide to make quizzes and exchange them with your classmates. Please have a first page that contains a form where the student must fill in their name, age and e-mail address. You must validate that the data here makes sense with PHP (i.e. they are strings/integers and the e-mail address is a proper e-mail address). Then students should be redirected to a short 5 question quiz to practice for the final exam and in the end their results shall be reported back to them. Make sure to display the correct answers to the questions they got wrong.

The web form must:

1. Include multiple choice and 'fill in the blanks' questions.
2. Include all of the following: button, radio button, check box, drop down box, text input.
3. The overall style should be professional. Put a bit of effort into appearance and aesthetics.
4. In the end, the scoring system should work. Nothing too fancy, but it should be an operational exam from start to finish.

## 10 Assignment 10 (Encryption and Search) –

### Problem 10.1 (SML Web Crawler)

A web crawler is a program that will store a copy (mirror) of a web site. Generally, crawlers 30pt 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.

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!">...  
 ...
```

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 writeFile(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;
```

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 10.2 (Parsing XML)

We are going to consider XML files with a special, more restricted format:

25pt

- Each line contains a Tag or a Content String.
- A Tag line is either of form “<NAME>” (for opening tags) or “</NAME>” (for closing tags). The lines between the opening and the closing tags can contain a Content String (in which case the Tag is called “leaf”) or several other tags (and then the Tag is named “node”).
- The Content String is a valid SML string.
- The file always starts with a Tag and ends with the corresponding closing Tag (this tag is called the “root” of the document).
- A Tag “B” that is contained in a node “A” is called “son” of “A”. Conversely, “A” is the “father” of “B”. “B” can have multiple “brothers” (which are Tags that also have “A” as a father).

An example of a valid XML file:

```
<first>
  <a2>
    <b3>
      text
    </b3>
    <b1>
      text3
    </b1>
  </a2>
</first>
```

The root is “first”, we have node “a2”, with leaves “b3” and “b1”.

We can “normalize” an XML file by alphabetically sorting the children of every node after the tag name. In case there are several children with the same name, we want them to keep the relative order they had in the input file (in other words, the sort is stable).

You are required to write an SML function `sortTags` that takes two arguments (the input file name and the output file name), reads the input file and then writes the normalized XML file to the output.

Example:

File "input.xml":	SML:	File "output.xml":
<pre>&lt;r&gt;   &lt;a3&gt;     &lt;b2&gt;       Father     &lt;/b2&gt;     &lt;b3&gt;       Mother     &lt;/b3&gt;   &lt;/a3&gt;   &lt;a1&gt;     &lt;ffx&gt;       AAA     &lt;/ffx&gt;     &lt;ff&gt;       BBB     &lt;/ff&gt;   &lt;/a1&gt; &lt;/r&gt;</pre>	<pre>sortTags('input.xml', 'output.xml');</pre>	<pre>&lt;r&gt;   &lt;a1&gt;     &lt;ff&gt;       BBB     &lt;/ff&gt;     &lt;ffx&gt;       AAA     &lt;/ffx&gt;   &lt;/a1&gt;   &lt;a3&gt;     &lt;b2&gt;       Father     &lt;/b2&gt;     &lt;b3&gt;       Mother     &lt;/b3&gt;   &lt;/a3&gt; &lt;/r&gt;</pre>

**Note:** The whitespace at the beginning of each line is ignored, but it should be preserved to the output file. Consider the input file to be well formatted (no closing tag / opening tag without match inside their parent tag).

---

### Problem 10.3 (Transport Layer Security)

15pt

- What is Transport-Layer Security (TLS)? Specify the steps which are executed by a client and a server in a TLS handshake.
- Explain the difference between public and private key encryption?
- What is an asymmetric key?
- Name 4 current commonly used encryption algorithms and explain one in detail.

### Problem 10.4 (Color Island)

On Color Island there are 13 blue chameleons, 15 yellow chameleons and 17 red chameleons. 20pt  
 When two chameleons of different colors meet, they change color into the third color. So for example, a blue and yellow chameleon meeting will result in two red chameleons. Is it possible that at some point all chameleons on the island have the same color? Write a formal description of this problem, as explained on the slides. What is one possible solution?

## 11 Assignment 11 (Astar-search) –

### Problem 11.1 (SML Web Crawler)

A web crawler is a program that will store a copy (mirror) of a web site. Generally, crawlers 25pt 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.

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!">...  
 ...
```

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 writeFile(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;
```

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!

---

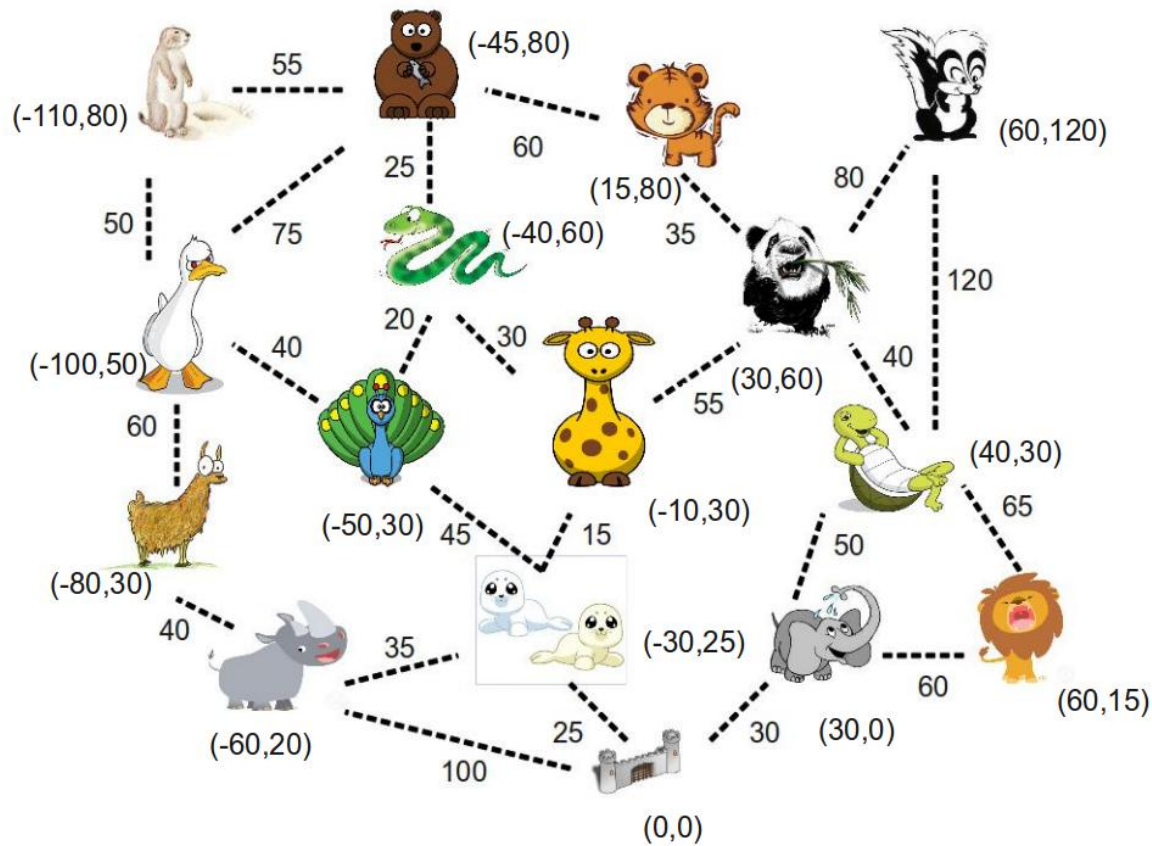
**Problem 11.2 ( $A^*$  search on Hamburg Zoo)**

Implement the  $A^*$  search algorithm in SML and test it on the following problem. You and your friends are visiting Hamburg Zoo. Not to get lost, you set up a few meeting points. You all start from the main gate, and the first area you get together is near the skunk. Next you decide to meet near the duck, while the last checkpoint is of course the main gate before leaving. 30pt

You may consider linear distance as heuristic. The lengths 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 rally point and the associated cost. Since we have three different rally points, you may write the output for each of them in comments or document it in a similar manner.





**Problem 11.3** Write 2 SML functions, *vis\_connected* and *dis\_connected* that, given an undirected(respectively directed) graph as a single argument, return *true* if the graph is connected and *false* otherwise. Both functions are supposed to use the DFS algorithm to find the solution. 30pt

The graph will be represented as a list of pairs of integers, where  $(a, b)$  is in the list if there is a path in the graph from  $a$  to  $b$ (be careful, in a directed graph the opposite does not hold).

**Problem 11.4 (Sudoku)**

15pt

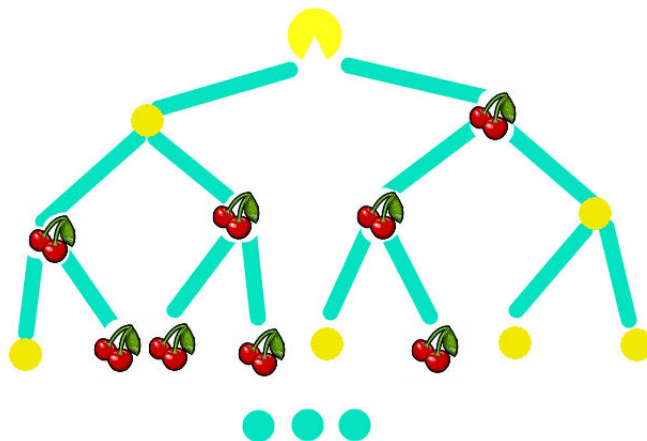
				8			4
	8	4		1	6		
			5			1	
1		3	8			9	
6		8				4	3
		2			9	5	1
		7		2			
			7	8		2	6
2			3				

This question will give you an excuse to play Sudoku (see [www.websudoku.com](http://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 11.5 (Binary Pacman)**

Consider the Pacman game played on a perfect binary tree (full binary tree in which all leaves are at the same depth) similar to the picture below. 25pt



In each node, there is the possibility of having Cherries or not. The goal of the game is to reach one of the leaves with the maximum number of collected Cherries. A valid move from a node is only to the children of that node.

To solve this problem, define:

- a cost function

- 2 admissible heuristics  $h_1(n)$  and  $h_2(n)$  which are different from  $h^*(n)$  – *the true cost from  $n$  to goal*, and not constant functions

that help find the solutions with most Cherries first, when using  $A^*$ . Give a short argument why your heuristics are admissible.

Prove that your choice of functions find the solutions with most Cherries first.

Provide a non-trivial example of the game with 5 levels and 15 Cherries randomly assigned to nodes (meaning don't put them all in the leaves, or all on 1 path, etc.) and show the order in which nodes are explored by  $A^*$  using each heuristic.