Name: Matriculation Number:

Final Exam
General CS I (320101)

December 14, 2010

You have two hours(sharp) for the test;
Write the solutions to the sheet.

The estimated time for solving this exam is 110 minutes, leaving you 10 minutes for revising your exam.

You can reach 55 points if you solve all problems. You will only need 48 points for a perfect score, i.e. 7 points are bonus points.

Different problems test different skills and knowledge, so do not get stuck on one problem.

	To be used for grading, do not write here														
prob.	1.1	1.2	2.1	2.2	2.3	3.1	3.2	4.1	4.2	5.1	5.2	6.1	6.2	Sum	grade
total	3	3	5	2	6	5	8	2	6	4	5	3	3	55	
reached															

Please consider the following rules; otherwise you may lose points:

- "Prove or refute" means: If you think that the statement is correct, give a formal proof. If not, give a counter-example that makes it fail.
- Always justify your statements. Unless you are explicitly allowed to, do not just answer "yes" or "no", but instead prove your statement or refer to an appropriate definition or theorem from the lecture.
- If you write program code, give comments!

1 Mathematical Foundations

3pt
5min

Problem 1.1 (Prime Induction)

Prove or refute that every positive integer $n > 1$ can be expressed as the product of one or more primes.

Note: The prime numbers in the final product are not necessarily distinct.

Solution: We will use strong induction to prove this.

Base case for $n = 2$: 2 is written as the product of one single prime number, 2.

Step case: Suppose that every integer between 2 and k can be written as the product of one or more primes. Now we show that $k+1$ can also be written as a product of primes. There are two cases:

- $k+1$ is prime, hence it is the product of one prime, i.e. itself.
- $k+1$ is not prime. Then $k+1 = ab$, where a, b are integers s.t. $1 < a, b < k+1$. By the induction hypothesis, $a = p_1 p_2 ... p_i$ where $p_l,\ l \in \{1, 2, .., i\}$ are prime factors. Similarly $b = q_1 q_2 .. q_j$ where $q_k,\ k \in \{1, 2, .., j\}$ are prime factors. Then $k+1$ can be expressed as $k+1 = p_1 p_2 ... p_i q_1 q_2 .. q_j$.

As $k+1$ is in the end written as a product of primes in all cases, we are done.

Problem 1.2 (Function Properties) 3pt 6min

Consider the function $f\colon \mathbb{R} \to \mathbb{R}$ with $f(x) = \begin{cases} x & \text{if } x \in \mathbb{Q} \\ -x+3 & \text{else} \end{cases}$

1. Prove or refute that function f is bijective on $\mathbb{R}$.
2. Compute $f \circ f$ and f^{-1}.

Solution:

1. First, we prove injectivity. Suppose $f(x) = f(y)$; we want to prove that this implies $x = y$.
   (a) $x, y \in \mathbb{Q}$; $f(x) = f(y) \Rightarrow x = y$.
   (b) $x, y \in \mathbb{R} \setminus \mathbb{Q}$; $f(x) = f(y) \Rightarrow -x+3 = -y+3 \Rightarrow x = y$.
   (c) It cannot happen $x \in \mathbb{Q}$ and $y \in \mathbb{R} \setminus \mathbb{Q}$ since that would imply equality of a rational number with an irrational one.
2. For this part, we use the fact that $x \in \mathbb{Q}$ iff $f(x) \in \mathbb{Q}$ and $x \in \mathbb{R} \setminus \mathbb{Q}$ iff $f(x) \in \mathbb{R} \setminus \mathbb{Q}$.

$$\begin{aligned}(f \circ f)(x) &= \begin{cases} f(x) & \text{if } f(x) \in \mathbb{Q} \\ -f(x)+3 & \text{else} \end{cases} \\ &= \begin{cases} x & \text{if } f(x) \in \mathbb{Q} \\ -(-x+3)+3 & \text{else} \end{cases} \\ &= \begin{cases} x & \text{if } f(x) \in \mathbb{Q} \\ x & \text{else} \end{cases}\end{aligned}$$

Thus, $(f \circ f)(x) = x$. From this, it follows that $f^{-1} = f$.

---

2 Abstract Data Types and Abstract Procedures

Problem 2.1 (Real Numbers ADT) 5pt 10min

Write an abstract data type for real numbers containing one significant digit (e.g. 2.1, -3.4, 5.0, -0.3 ...) considering the following requirements:

1. there is exactly one digit before and after the dot
2. real numbers can be positive or negative; you can consider 0.0 to be positive

Show the ground constructor terms for the following real numbers: 3.5, -2.6, 0.0, 0.4

Hint:
Consider a special sort for representing digits $0 - 9$

Solution: Consider sort u$\mathbb{R}$ for unsigned real numbers, $\mathbb{R}$ for signed real numbers and $\mathbb{D}$ for digits. The simplest way to represent digits is to have constructors for all of them (10 constructors), while another idea would be to have a modified version of unary natural numbers $\mathbb{N}$:

1. $$\langle\{u\mathbb{R},\mathbb{R},\mathbb{D}\},\{[0_d\colon \mathbb{D}],[1_d\colon \mathbb{D}],[2_d\colon \mathbb{D}],[3_d\colon \mathbb{D}],[4_d\colon \mathbb{D}],[5_d\colon \mathbb{D}],[6_d\colon \mathbb{D}],[7_d\colon \mathbb{D}],[8_d\colon \mathbb{D}],[9_d\colon \mathbb{D}],[neg\colon u\mathbb{R}\to\mathbb{R}],$$

2. $3.5 = pos(dot(3_d, 5_d))$
   $-2.6 = neg(dot(2_d, 6_d))$
   $0.0 = pos(dot(0_d, 0_d))$
   $0.4 = pos(dot(0_d, 4_d))$

Problem 2.2 (Substitutions) 2pt 8min

1. Apply the substitutions

$$\sigma := [f(a)/x], [g(a, f(a), b)/z], \qquad \tau := [a/x], [h(b, c)/y], [c/z]$$

   to the terms $s := g(x, f(y), z)$ and $t := h(f(x), g(y, z, x))$

2. Prove or refute that substitution composition is commutative (i.e. $\sigma_1 \circ \sigma_2 = \sigma_2 \circ \sigma_1$).

Solution:

1. Applying the substitutions:
   $\sigma(s) = g(f(a), f(y), g(a, f(a), b))$
   $\sigma(t) = h(f(f(a)), g(y, g(a, f(a), b), f(a)))$
   $\tau(s) = g(a, f(h(b, c)), c)$
   $\tau(t) = h(f(a), g(h(b, c), c, a))$

2. To check that we apply the two substitutions in diffrent order on the term s:
   $\sigma(\tau(s)) = g(a, f(h(b, c)), c)$
   $\tau(\sigma(s)) = g(f(a), f(h(b, c)), g(a, f(a), b))$
   Obviously the two expressions are different. Therefore substitution application is not commutative.

Problem 2.3 (A Reindeer Lottery) 6pt 12min

Soon after the last Christmas, Rudolf retired, so Santa has to decide which of his reindeers will be the navigator of the sledge. To do so, he needs to find the "special number" of each reindeer based on their names.

Santa first converts every letter of the names to a unary natural number ("A" becomes o, "B" becomes $s(o)$ and so on) and then asks your help to solve the following algorithm: start at the first position of the converted word (considered position 0), and then sum up all the values in the order obtained by jumping from position i to position $(i+2) \mod L$ (where L is the length of the name); you should stop only when you have used all the converted letters of the word.

For example, consider VIXEN the reindeer: his special number will be determined by suming the corrseponding codes in the order V-X-N-I-E.

1. You are required to write an abstract procedure that returns the special number of a reindeer, using the following ADT:

$$\langle\{\mathbb{N},\mathbb{L}\},\{[nil\colon\mathbb{L}],[con\colon\mathbb{N}\times\mathbb{L}\to\mathbb{L}],[o\colon\mathbb{N}],[s\colon\mathbb{N}\to\mathbb{N}]\}\rangle$$

   The argument word is given as a list of unary natural numbers, each representing the number associated to the letter (Santa took care of this).

2. Does the procedure always terminate? If not, give an example of such a name for which the procedure does not terminate!

Hint:

- You can define helper procedures!
- An abstract procedure terminates if all the "helper" procedures it uses also terminate for any input.

Solution: One can observe that the abstract procedure that goes from i to $(i+2) \mod L$ and passes through all the L letters terminate only for odd word lengths (odd L). This happens because for even L, the procedure will pass through positions 0, 2,... $L-2$ and then cycles back to 0.

Therefore, we will define a helper procedure for word length, unary number addition, remainder for division by 2 (used for determining whether L is odd or even):

- $\langle len{::}\mathbb{L}\to\mathbb{N}\,;\,\{len(nil)\rightsquigarrow o, len(con(x_{\mathbb{L}}))\rightsquigarrow s(len(x_{\mathbb{L}}))\}\rangle$
- $\langle add{::}\langle\mathbb{N},\mathbb{N}\rangle\to\mathbb{N}\,;\,\{add(x_{\mathbb{N}},o)\rightsquigarrow x_{\mathbb{N}}, add(x_{\mathbb{N}},s(y_{\mathbb{N}}))\rightsquigarrow s(add(x_{\mathbb{N}},y_{\mathbb{N}}))\}\rangle$
- $\langle mod2{::}\mathbb{N}\to\mathbb{N}\,;\,\{mod2(o)\rightsquigarrow o, mod2(s(o))\rightsquigarrow s(o), mod2(s(s(x_{\mathbb{N}})))\rightsquigarrow mod2(x_{\mathbb{N}})\}\rangle$

Now, the helper procedure should cycle for even length (in order to properly emulate the behaviour described in the problem) and return the sum of all the values in the list for odd length. Suppose that the length of the list (or more precisely the length modulo 2) is already given as a parameter. Thus we have:

$$\langle h{::}\langle \mathbb{L}, \mathbb{N}\rangle \to \mathbb{N}\,;\,\{h(x_{\mathbb{L}}, o) \rightsquigarrow h(x_{\mathbb{L}}, o), h(nil, s(o)) \rightsquigarrow o, h(cons(a_{\mathbb{N}}, x_{\mathbb{L}}), s(o)) \rightsquigarrow add(a_{\mathbb{N}}, h(x_{\mathbb{L}}, s(o)))\}\rangle$$

It is obvious that if the second parameter is o, the procedure above will never terminate. Finally, the special number procedure:

$$\langle special{::}\mathbb{L} \to \mathbb{N}\,;\,\{special(x_{\mathbb{L}}) \rightsquigarrow h(x_{\mathbb{L}}, mod2(len(x_{\mathbb{L}})))\}\rangle$$

---

3 Programming in Standard ML

Problem 3.1 (Secret Message cont'd) 5pt 10min

Some time ago, a couple of students at GenCS wanted to develop a way to communicate with each other in text that makes it impossible for others to understand. Thus they thought of encoding strings as integers or *keys* (greater or equal than zero and less than ten): each string from a list (the list of strings has the same length as the list of integers) gets an integer from the *keys* list. Since they were very good with SML, they implemented two functions:

```
val secretEncodeList = fn : string list * int list -> (string * int) list
val getEncodingStr = fn : int * (string * int) list -> string
```

The first one (secretEncodeList) creates a list of pairs (string,int) to be used for decoding and the second one (getEncodingStr) takes an int (*key*) and gives its string representation from the encoding list. See below an example of their functionality:

```
- secretEncodeList(["He","o","ll","␣w", "rl","d!"], [4,3,5,1,0,2]);
val it = [("He",4),("o",3),("ll",5),("␣w",1),("rl",0),("d!",2)]
  : (string * int) list
- getEncodingStr(0, it);
val it = "rl" : string
```

Your task:

Now the students want you to write a function to decode such a message which actually is an integer (without leading zeros) whose digits are supposed to be the *keys* in the encoding. You are allowed to use their functions (consider them implemented). The function signature is described below and you can also see an example:

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

Example:

```
- decodeMessage(4531302, ["He","o","ll"," w", "rl","d!"], [4,3,5,1,0,2]);
val it = "Hello world!" : string
```

Note: The resulting encoding pair list is a bijective function (strings and integers are distinct).

Solution:

```
fun myimplode([], str) = implode(str)
  | myimplode(s::strs, str) = myimplode(strs, str@explode(s));

fun decode_helper(0, encoding, res) = myimplode(res, [])
  | decode_helper(x, encoding, res) =
                          let val digit = x mod 10 in
                   decode_helper(x div 10, encoding, getEncodingStr(digit, encoding)::res)
               end;


fun decodeMessage(x:int, alphabet:string list, keys:int list) =
                          let val encoding = secretEncodeList(alphabet, keys) in
                   decode_helper(x, encoding, [])
               end;
```

8pt
Problem 3.2 (Prime Differences) 16min

A list is called a Prime Difference List if and only if the difference (in absolute value) between any two neighbors (consecutive elements of the list) is a prime number.

1. Write an SML function which generates all the permutations of a list.
   val GeneratePermutations = fn : int list −> int list list

2. Write an SML program which takes as input the list of integers and outputs a permutation of that list which is a Prime Difference List. You may use the function GeneratePermutations even if you have not succeeded in the previous task.
   val PrimeDifferenceList = fn : int list −> int list

Example:

PrimeDifferenceList([3,2,5]) = [2,5,3];

This would be one of the valid outputs, as the difference between 2 and 5 is 3 and between 5 and 3 is 2.

Note: You do not need to handle the cases where the input does not have any PDL permutation.

Solution: The idea is pretty straight-forward. First all the permutations are generated then we test until we find one that is viable. Here is the code for the first task:

```
fun PrePend(x, [] ) = []
  | PrePend(x,h::t) = (x::h)::PrePend(x,t);

fun RemPos(h::t, 0) = t
  | RemPos(h::t, x) = h::RemPos(t,x−1);

fun GenAux(h::[], _) = [[h]]
  | GenAux(h::t , 0) = PrePend(h, GenAux(t,length(t)−1))
  | GenAux(h::t , x) = PrePend(List.nth(h::t, x), GenAux(RemPos(h::t, x),length(t) − 1))
                             @ GenAux(h::t, x−1)

fun GeneratePermutations(x) = GenAux(x, length(x)−1);
```

Here is the code for testing if an integer is a prime:

```
fun IsPrimeAux(x, y) = if Real.fromInt(y) > Math.sqrt(Real.fromInt(x))
                       then true
                       else if x mod y = 0
                            then false
                            else IsPrimeAux(x, y+1);

fun IsPrime(0) = false
  | IsPrime(1) = false
  | IsPrime(x) = IsPrimeAux(x,2);
```

And the main functions:

```
fun IsViable(ls) = if length(ls) < 2
                   then true
                   else let
                            val h1::h2::t = ls
                        in
                            IsPrime(abs(h1−h2)) andalso IsViable(h2::t)
                        end;

fun FindPrimeDiffSet(h::t) = if IsViable(h)
                             then h
                             else FindPrimeDiffSet(t);

fun PrimeDifferenceList(ls) = let
                                val possibilities = GeneratePermutations(ls)
                              in
                                FindPrimeDiffSet(possibilities)
                              end;
```

---

4 Formal Languages and Codes

Problem 4.1 (Greek Codes) 2pt 4min

The Greeks developed a very ingenious way of transmitting messages by the use of their alphabet. The character code that maps the Greek letters to symbols from the ASCII table is described below (the Greek letters are represented by their names instead of their symbol)

alpha	beta	gamma	delta	epsilon	zeta	eta	theta
hr	:)	s!	ma	ris	Me	ry	y_
iota	kappa	lambda	mu	nu	xi	omicron	pi
ew	a_	ar	_H	Ha	s!	A	!
rho	sigma	tau	upsilon	phi	chi	psi	omega
_Y	pp	He	ida	e	y	ol	Ha

Using the extension of this code encode the following message

$$\nu\sigma\chi\mu\psi\upsilon\chi\xi$$

Solution: $c(\nu\sigma\chi\mu\psi\upsilon\chi\xi)$ = Happy Holidays!

Problem 4.2 (Globes) 6pt 12min

Consider that you have g globes, each built by putting together exactly p parts. Also let there be c colors available in total for constructing these globes. The purpose is to build an injective function, such that colors are assigned to each part individually, before making actual globes. Globe g_i is composed of parts p_{ij}, where $i \in \{1, 2, .., g\}$ and $j \in \{1, 2, .., p\}$. Here is an example:

Let $g = 2$, $p = 2$, $c = 3$. Consider the map $f : p \mapsto c^+$ as follows: $f(p_{11}) = c_1$, $f(p_{12}) = c_1c_2$, $f(p_{21}) = c_1c_3$, $f(p_{22}) = c_2c_1c_3$. So here we have four parts that will compose in the end two globes and for example globe g_1 will have the colors c_1 and c_2. Also, note that the total number of painted colors is $1 + 2 + 2 + 3 = 8$.

1. For the case $g = 3$, $p = 2$, $c = 3$, what is the minimum number of total painted colors (the equivalent of the 8 above) on the parts, such that we are sure that each constructed globe has all colors on it? A globe is made here by putting 2 random parts together, since $p = 2$.
2. Provide a general formula for the same minimal number for the case in which $g < \lfloor \frac{c^2}{2} \rfloor$. Hint: this is to help you, not to complicate the problem.
3. Are the above constructions character codes? Are their extensions string codes? You can only consider the first part if you have not solved the second.

Solution:

1. We can construct the following function: $f(p_{11}) = c_1c_2$, $f(p_{12}) = c_2c_3$, $f(p_{21}) = c_1c_3$, $f(p_{22}) = c_1c_2c_3$, $f(p_{31}) = c_2c_1c_3$, $f(p_{32}) = c_3c_2c_1$.

   We used colors 15 times. Also notice that in whatever way we choose p_{ij} and p_{lk}, where $\{i, j, k, l\} \in \{1, 2, 3\}$, their concatenation will definitely contain all three colors.
2. The point of the restriction is that it allows us to choose c as maximum length of each color string. Thus we will have g parts that can be colored like in the example above in $c - 1$ long color strings and the rest of $g \cdot (p - 1)$ can have colors of length c each. Hence the final formula is $g \cdot (c - 1) + g \cdot c \cdot (p - 1) = g \cdot (cp - 1)$.
3. Due to injectivity of the function, f is a character code, however since by this construction we encounter prefixes, the extension will not be a string code.

---

5 Boolean Algebra

Problem 5.1 (Playing with Distributivities) 4pt 8min

Prove or refute the following equivalences, using the laws of Boolean Algebra:

1. $a + \overline{a} * b = a + b$
2. $(a * b + b * c) + c * a = ((a + b) * (b + c)) * (c + a)$

Solution: Each equivalence makes use in the proof of the distributivity of multiplication with respect to addition: a+b*c=(a+b)*(a+c).

1.

$$\begin{aligned} a + \neg a * b &= (a + \neg a) * (a + b) \\ &= 1 * (a + b) = a + b. \end{aligned}$$

2.

$$\begin{aligned} a * b + b * c + c * a &= a * b + (b * c + c * a) = (a + b * c + c * a) * (b + b * c + c * a) \\ &= (a * (1 + c) + b * c) * (b * (1 + c) + c * a) \\ &= (a + b * c) * (b + c * a) = (a + b) * (a + c) * (b + c) * (b + a) \\ &= (a + b) * (b + c) * (c + a). \end{aligned}$$

Problem 5.2 (QMC application) 5pt 10min

Execute Quine-McCluskey algorithm to get the minimum polynomial for the function with the provided truth table:

x_1	x_2	x_3	x_4	f
T	T	T	T	T
T	T	T	F	F
T	T	F	T	T
T	T	F	F	F
T	F	T	T	T
T	F	T	F	F
T	F	F	T	T
T	F	F	F	F
F	T	T	T	T
F	T	T	F	F
F	T	F	T	F
F	T	F	F	F
F	F	T	T	T
F	F	T	F	F
F	F	F	T	F
F	F	F	F	F

Solution:

QMC_1 :

x_1	x_2	x_3	x_4
T	T	T	T
T	T	F	T
T	F	T	T
T	F	F	T
F	T	T	T
F	F	T	T

x_1	x_2	x_3	x_4
T	T	X	T
T	X	T	T
X	T	T	T
T	X	F	T
T	F	X	T
X	F	T	T
F	X	T	T

x_1	x_2	x_3	x_4
T	X	X	T
X	X	T	T

Therefore the prime implicants are $x_1\, x_4$ and $x_3\, x_4$

QMC_2 :

	TTTT	TTFT	TFTT	TFFT	FTTT	FFTT
$x_1\, x_4$	T	T	T	T	F	F
$x_3\, x_4$	T	F	T	F	T	T

Therefore both prime implicants are essential.

Final result: $f = x_1\, x_4 + x_3\, x_4$

6 Propositional Logic

Problem 6.1 (Tableau Calculus) 3pt 6min

1. Explain when we call a formula unsatisfiable and give an example.
2. Use the tableau refutation method to prove the validity of the formula

$$(X \Rightarrow Y) \Rightarrow (\neg Y \Rightarrow \neg X)$$

Solution:

1. A Boolean expression E is called unsatisfiable iff it evaluates to F for all variable assignments φ, i.e. $I_\varphi(E) = F$.

   An example for such an expression is $x \wedge \neg x$ (many other possibilities here)

2. Applying the tableau method:

$$\begin{array}{c} ((X \Rightarrow Y) \Rightarrow (\neg Y \Rightarrow \neg X))^F \\ (X \Rightarrow Y)^T \\ (\neg Y \Rightarrow \neg X)^F \\ X^F \vee Y^T \\ Y^F \\ X^T \\ X^F | \; Y^T \\ \bot \mid \bot \end{array}$$

   Therefore there is no model. Thus the initial expression is unsatisfiable.

   Therefore the expression $(X \Rightarrow Y) \Rightarrow (\neg Y \Rightarrow \neg X)$ is valid.

Problem 6.2 (Hilbert Calculus) 3pt 6min

Consider a calculus given by the axioms

1. $\mathbf{K} := P \Rightarrow Q \Rightarrow P$,
2. $\mathbf{S} := (P \Rightarrow Q \Rightarrow R) \Rightarrow (P \Rightarrow Q) \Rightarrow P \Rightarrow R$

and the following 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 that $(M \Rightarrow N) \Rightarrow M \Rightarrow M$

Solution:

1. Applying the substitutions $[M/P], [N/Q], [M/R]$ to axiom **S** yields

$$(M \Rightarrow N \Rightarrow M) \Rightarrow (M \Rightarrow N) \Rightarrow M \Rightarrow M$$

2. Applying the substitutions $[M/P], [N/Q]$ to axiom **K** yields $M \Rightarrow N \Rightarrow M$
3. Applying MP on 1. and 2. gives $(M \Rightarrow N) \Rightarrow M \Rightarrow M$
