Name: Matriculation Number:

Final Exam
General CS II (320102)

May 23., 2015

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

The estimated time for solving this exam is 112 minutes, leaving you 8 minutes for revising your exam.

You can reach 107 points if you solve all problems. You will only need 100 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	3.1	3.2	3.3	4.1	4.2	4.3	5.1	6.1	Sum	grade
total	10	10	15	12	15	8	3	4	8	12	10	107	
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 Graphs & Trees

Problem 1.1 (Edges of Connected Graphs)

Show that a graph G with n vertices is connected if it has more than $\frac{(n-1)(n-2)}{2}$ edges. 10pt

Hint: Consider the highest number of edges a graph can have without being connected. 10min

Solution: The highest number of edges a graph can have without being connected. It must have two connected components, and, to maximize the number of edges, they must be size $n-1$ and 1. To maximise the edges, the large component must be a complete graph, which will have $(n-1)(n-2)/2$ edges.

A strict translation would go like:

Suppose that G is not connected. Then it has a component of k vertices for some k in range of 1 to $n-1$. The most edges G could have is $C(k,2)+C(n-k,2) = k^2 - nk + (n^2-n)/2$. This quadratic function of f is minimized at $k = n/2$ and maximized at $k = 1$ or $k = n-1$. Hence, if G is not connected, the number of edges does not exceed the value of this function at 1 and at $n-1$, namely, $(n-1)(n-2)/2$.

Or by induction: The solution by induction on the number of vertices n only needs to observe that the induction step only needs $n-1$ more edges when a vertex is added to the set of n vertices from inductive hypothesis.

Problem 1.2 (Balanced Trees)

10pt

10min

1. Implement an SML data type for binary trees and
2. a function that checks whether a binary tree is balanced.
3. Explain How would you change the data type and algorithm to allow for a ternary tree?

Solution:

```
int maxDepth(TreeNode root) {
   if (root == null) {
            return 0;
    }
    return 1 + Math.max(maxDepth(root.left), maxDepth(root.right));
}

int minDepth(TreeNode root) {
    if (root == null) {
        return 0;
    }
    return 1 + Math.min(minDepth(root.left), minDepth(root.right));
}

boolean isBalanced(TreeNode root){
    return (maxDepth(root) − minDepth(root) <= 1);
}
```

2 Circuits and Positional Number Systems

Problem 2.1 (Digit Display)

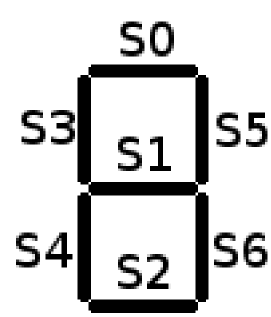
15pt

15min

Suppose that you are given a set of 4 binary inputs that represent a digit (0-9) in binary (from ’0000’ for 0 to ’1001’ for 9).

Consider the segments in the order described in the image below:

For example, one could, for the digit 1, turn on segments $S3$ and $S4$, so only the outputs corresponding to these segments will be 1 if the input is ’0001’.



1. Create a mapping between the digits and the states of the segments.
2. Write the truth tables for segments $S1$ and $S3$.
3. Derive the minimal polynomials for them.

4. Draw one circuit with two outputs representing the states of the two segments.

Solution: For each digit, one has to determine what segment wil be on. For example, the digit 2 should turn on segments $S0$, $S5$, $S1$, $S4$ and $S2$. The final implementation is dependant on the choice of digit-display, but for example one can use segment $S1$ in digits 2 (0010), 3 (0011), 4 (0100), 5 (0101), 6 (0110), 8 (1000), 9 (1001). Thus we can create the following truth table:

i_1	i_2	i_3	i_4	$S1$
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1

The next steps would be applying QMC to obtain the minimum polynomial and then implement it in a circuit:

$$\overline{i_1}\,\overline{i_2}\,i_3 + \overline{i_1}\,i_2\,\overline{i_3} + i_1\,\overline{i_2}\,\overline{i_3} + \overline{i_1}\,i_3\,\overline{i_4}$$

The same procedure should be done for every segment.

3 Machines

Problem 3.1 Write an ASM program that computes the nth 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. 12pt 12min

Problem 3.2 (Perfect Squares in $\mathcal{L}$(`VMP`)) 15pt 20min

Write an $\mathcal{L}$(`VMP`) procedure that, given a number n as the only parameter, returns 1 if n is a perfect square and 0 otherwise.

(Recall: an integer n is a perfect square if there exists another integer x such that $n = x \cdot x$.)

Additionally, please provide the same procedures in μML as well.

Hint: You may want to define additional procedures.

Solution:

```
proc 2 32,                                  // check(a,b) for checking (a == b*b) -> call by 'call 0'
arg 1, arg 2, arg 2, add, leq, cjp 15,      // if(2b > a) jump to RETURN0
arg 2, arg 2, mul, arg 1, sub, cjp 15,      // if(b*b == a) jump to RETURN1

con 1, arg 2, add, arg 1, call 0, return,   // otherwise we go on incrementing b
                                            // we call check(a, b+1)
con 0, return,                              // RETURN0 -> a is not a perfect square
con 1, return,                              // RETURN1 -> a is a perfect square

proc 1 33,                                  // isSquare(a) -> to be called by call 'call 32'
arg 1, cjp 25,                              // if(a == 0) jumps to RETURN_1;
con 1, arg 1, sub, cjp 18,                  // if(a == 1) jumps to RETURN_1;
arg 1, con 0, leq, cjp 8,                   // if(0 > a) jumps to RETURN_0;
con 2, arg 1, call 0                        // here a >= 2; calling check(a, 2)

con 0, return,                              // RETURN_0 -> a is not a perfect square
con 1, return,                              // RETURN_1 -> a is a perfect square


con 9, call 32, halt                        // calling it on 9
```

Problem 3.3 (TM compare two numbers)

Given the alphabet $\{0, 1, \#\}$, where $\#$ symbolizes an empty cell, consider a tape with the input 0^n1^m, where n and m are natural numbers, (followed by infinitely many $\#$s). Design a TM that halts in a state "yes" if $n > m$ and in state "no" otherwise. 8pt 8min

Solution: Firstly, it is possible to come up with a solution that covers the case $n = m = 0$, but in complexity theory we are more interested in how TM act on "longer" inputs, so this case may as well be disregarded. (i.e. don't subtract points for that).

Missing entries in the transition table can be filled arbitrarily; i.e. they are irrelevant for the solution of the problem.

Old	Read	Write	Move	New
s_0	0	$\#$	right	s_1
s_0	1	1	stop	"no"
s_0	$\#$	$\#$	stop	"no"
s_1	0	0	right	s_1
s_1	1	1	right	s_1
s_1	$\#$	$\#$	left	s_2
s_2	1	$\#$	left	s_3
s_2	0	0	stop	"yes"
s_3	1	1	left	s_3
s_3	0	0	left	s_3
s_3	$\#$	$\#$	right	s_0

4 Internet/WWW/XML

Problem 4.1 (Information units)

Write down 8 units of information in increasing order of capacity. 3pt

Solution: *bit < byte < kilobyte < megabyte < gigabyte < terabyte < petabyte < exabyte* 3min

Problem 4.2 (WWW Nomenclature)

4pt 4min

1. What does the acronym HTML stand for?
2. What organizations make the Web standards? Name two.
3. What is HTML tag for the largest heading?
4. What is the correct HTML tag for inserting a line break?

Problem 4.3 (Key Exchange)

10pt

Discuss the purpose of the Diffie/Hellmann key exchange algorithm and explain how it works (conceptually). If you use colors to explain, also say what math functions could be used for the real application. 10min

5 Problem Solving and Search

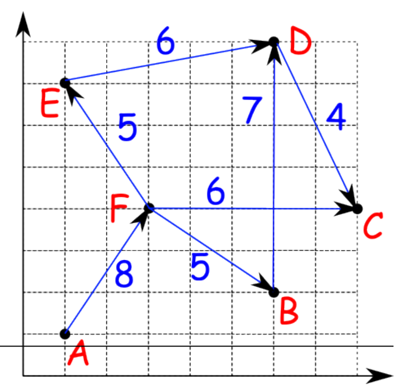
Problem 5.1 (A* on Cartesian Grid)

12pt

12min

You are given the following set of points (nodes) on a Cartesian grid: $A(1,1)$, $B(6,2)$, $C(8,4)$, $D(6,8)$, $E(1,7)$, $F(3,4)$, and the following set of edges between them: $(A,F,8)$, $(F,B,5)$, $(F,C,6)$, $(F,E,5)$, $(E,D,6)$, $(D,C,4)$, $(B,D,7)$. The initial node is A, the goal is C.

1. Design an admissible heuristic to be used for an A* algorithm for the given problem. Give the value of the heuristic applied on every node. You **do not** have to prove that it is admissible.
2. Using the heuristic stated before, write down the order of access of the nodes, when A* strategy is used.



Solution:

- The easiest-to-use heuristic is the straight-line distance (we can observe that the length of the edges is always greater or equal than the planar distance between the points). We get:

$$\begin{aligned} h(A) &= \sqrt{58} \\ h(B) &= \sqrt{26} \\ h(C) &= \sqrt{0} \\ h(D) &= \sqrt{20} \\ h(E) &= \sqrt{58} \\ h(F) &= \sqrt{5} \end{aligned}$$

- Using this heuristic, we will visit the nodes in the following order:

$$A, F, C$$

6 Programming in Prolog

Problem 6.1 (Relationships)

Given the following statements write them in Prolog to write a program that determines whether Sansa is Rickon's sister and whether Lyarra is Arya's ancestor. Explain how your query works. 10pt

10min

1. Lyarra is Eddard's mother
2. Bran is Arya's brother
3. Bran is Rickon's brother
4. Arya is Sansa's sister
5. Eddard is Arya's father

Solution:

```
mother(lyarra, eddard).
brother(bran, arya).
brother(bran, rickon).
sister(arya, sansa).
father(eddard, arya).

sibling(X, Y) :- brother(X,Y);
    brother(Y,X);
    sister(X,Y);
    sister(Y,X);
    sibling(X,Z), sibling(Y,Z).

parent(X,Y) :- father(X,Y); mother(X,Y).

ancestor(X,Y) :- parent(X,Y);
    parent(X,Z), ancestor(Z,Y).

?- sibling(sansa, rickon).
?- ancestor(lyarra, arya).
```