Name: Matriculation Number:

Final Exam Spring 2011
General CS II (320201)

May 24., 2011

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

The estimated time for solving this exam is 115 minutes, leaving you 5 minutes for revising your exam.

You can reach 58 points if you solve all problems. You will only need 53 points for a perfect score, i.e. 5 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	1.3	1.4	2.1	2.2	2.3	3.1	3.2	4.1	5.1	Sum	grade
total	4	3	3	7	6	5	5	8	7	5	5	58	
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, Circuits and Positional Number Systems

4pt
8min

Problem 1.1 (Hamiltonian Hypercube)

An $n-$cube is a cube in n dimensions. For $n = 1$ we have a line segment, for $n = 2$ a square is obtained, for $n = 3$ the result is a standard cube, etc. In general, going to the $(n+1)^{th}$ dimension translates to creating a copy of the $n-$cube $C = (V, E)$. Let the copy be named $C' = (V', E')$. The $(n+1)-$cube is obtained when connecting each vertex $V_i \in V$ to its corresponding $V_i' \in V'$ by one edge. This new figure represents the cube in the $(n+1)^{th}$ dimension. Going from two dimensions to three dimensions would be a good example to try out the procedure.

If we consider the cube to be composed of the vertices and edges only, show that every $n-$cube has a Hamiltonian circuit. You should also describe how it can be obtained in a detailed manner.

Note: Recall that a Hamiltonian path is a path in an undirected graph that visits each vertex exactly once. A Hamiltonian cycle (or Hamiltonian circuit) is a cycle in an undirected graph which visits each vertex exactly once and also returns to the starting vertex.

Solution: We can proceed by induction. As base case, consider $n = 1$. We visit each vertex of a two-vertex graph with an edge connecting them.

Now assume the statement holds for dimension k. To build a $(k+1)-$cube, we take two copies of the $k-$cube and connect the corresponding vertices. Take the Hamiltonian circuit on one cube and reverse it in the other. Then choose an edge on one that is part of the circuit and the corresponding edge on the other and delete them from the circuit. Finally, add to the path connections from the corresponding endpoints on the cubes which will produce a circuit on the $(k+1)-$cube.

Problem 1.2 (TCN operations) 3pt
6min

Let $A = NNN$ and $B = DD$ two decimal numbers, where NNN are the first 3 digits of your matriculation number and DD is your birth date (e.g. 15. August $\rightsquigarrow$ $DD = 15$, 3. September $\rightsquigarrow$ $DD = 03 = 3$, i.e. discard the leading zero).

1. convert the numbers into an n-bit TCN and determine the minimal n needed for encoding A and B.
2. perform a binary subtraction $A - B$ and check the result by converting back to the decimal system. Show your steps.

Solution: Different possible solutions, take for example $A = 203$, $B = 15$
n should be 9. The 9-bit TCN representations are:

$$\langle\!\langle B(203) \rangle\!\rangle = 011001011$$

$$\langle\!\langle B(15) \rangle\!\rangle = 000001111$$

The subtraction $A - B$ in TCN representation is

$$\langle\!\langle B(203) \rangle\!\rangle - \langle\!\langle B(15) \rangle\!\rangle = \langle\!\langle B(203) \rangle\!\rangle + \langle\!\langle \overline{B(15)} \rangle\!\rangle + 1$$

This corresponds to

$$011001011 - 000001111 = 011001011 + 111110000 + 1 = 010111100$$

In fact: $\langle\!\langle B(203 - 15) \rangle\!\rangle = \langle\!\langle B(188) \rangle\!\rangle = 010111100$

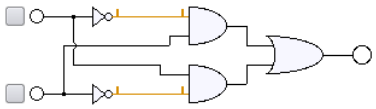
Problem 1.3 (Linked Switches) 3pt
6min

You have a big room and a switch in each corner (4 switches). Each such switch can be in ON or OFF state. There is an electrical circuit which connects the 4 switches to the light system of the room.

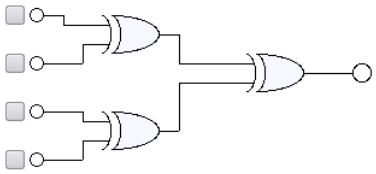
Task: Design, using AND, OR and NOT gates, a circuit with 4 inputs (the state of the 4 switches) and 1 output (the state of the light system) in such a way that switching any of the corners independently will cause the light system to switch.

Solution: The obvious choice is to use XOR gates and XOR all the outputs. As such, a switch in any of the corners will change the parity and hence the state of the light system.

The XOR gate can easily be built like this:



Now, using the XOR gates, the solution is straight forward:



Problem 1.4 (Magnitude Comparator) 7pt 14min

1. Prove the following equivalence:

$$xy + \overline{x}\,\overline{y} = \overline{x \oplus y} = NOR(x\overline{y}, \overline{x}y)$$

   What can be deduced about the relationship between x and y from this equation?
2. Your task is to create a circuit for a 4-bit magnitude comparator.
   - It has two 4-bit binary numbers as inputs $A = A_3A_2A_1A_0$ and $B = B_3B_2B_1B_0$. It has 3 output lines: E (for $A = B$), G (for $A > B$), and L (for $A < B$). As a result of the magnitude comparison one of the output lines should be set to 1, and the other two - to 0.
   - For each of the three possible outputs, briefly explain what conditions are needed so that it is set to 1. You may give the Boolean expression if you wish, but this is not obligatory.
   - Then draw the circuit according to your observations.
   - You may use any type of logic gates (AND, OR, negation, NAND, NOR, XOR), n-ary gates are also allowed.

Solution:

- The proof can be done with a simple truth table:

x	y	$xy + \overline{x}\,\overline{y}$	$\overline{x \oplus y}$	$NOR(x\overline{y}, \overline{x}y)$
0	0	1	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	1	1

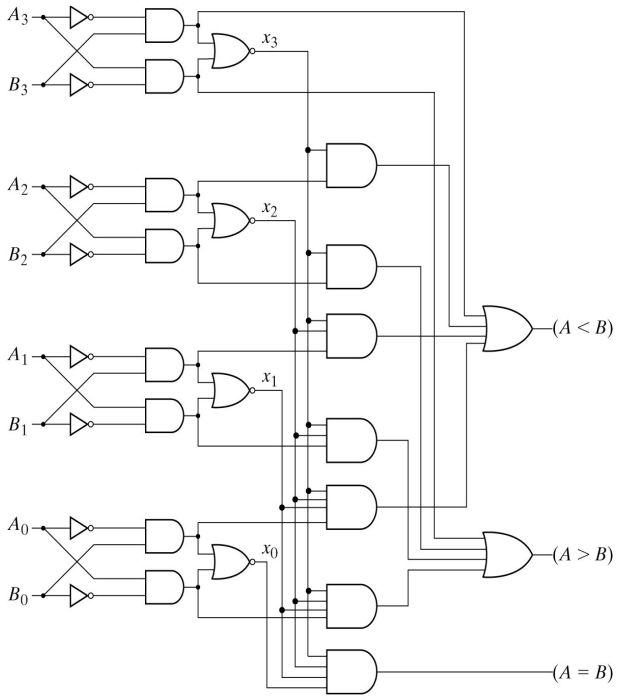
  The expressions can be used to check whether two bits are equal.

- Two 4-bit numbers would be equal if $A_i = B_i$ for all i. So we have to use one of the versions of the above expression for each of the four pairs of bits, obtaining E_3, E_2, E_1, and E_0. We need all of them to be 1, so at the end we will use an AND gate to obtain E.

  $A > B$ if $A_3 > B_3$. If $A_3 < B_3$, then A is not bigger. But if the first bits are equal, we have to compare the second bits. $A_3 > B_3$ can be checked with $A_3\overline{B_3}$. Then, if this is not true, we have to still check with the next bits, so we will also use one additional variable from E, to verify that the previous bits were equal. I.e. for A_2 and B_2 we will have $E_3A_2\overline{B_2}$. And so on, at the end at least one of the 4 bit comparisons has to give 1, so we OR the results. The final Boolean expression is $A_3\overline{B_3} + E_3A_2\overline{B_2} + E_3E_2A_1\overline{B_1} + E_3E_2E_1A_0\overline{B_0}$.

  For $A < B$ the same reasoning applies, just everything is mirrored. So the corresponding Boolean expression is $\overline{A_3}B_3 + E_3\overline{A_2}B_2 + E_3E_2\overline{A_1}B_1 + E_3E_2E_1\overline{A_0}B_0$.

  The final circuit is:


A_3
B_3
x_3
A_2
B_2
x_2
A_1
B_1
x_1
A_0
B_0
x_0
$(A < B)$
$(A > B)$
$(A = B)$

2 Machines

6pt
12min

Problem 2.1 (How many leaves?)

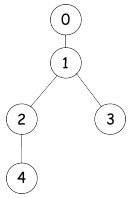
In a register machine you are given a memory state that represents a tree $\mathcal{T}$ labeled with numbers between 0 and $n-1$. You will find the number n of nodes in $\mathcal{T}$ in $P(0)$ and for each node x in $\mathcal{T}$ the parent of node x in $P(x+2)$ (the root of the tree is always labeled with 0 and $P(2) = -1$). That leaves $P(1)$ free.

Your task is to write an `ASM` program that, given the memory layout above, counts the number of leaves that the tree has and stores it in $P(1)$.

Example:

$P(i)$	5	**2**	-1	0	1	1	2
i	0	**1**	2	3	4	5	6

represents



The root is node 0. There are two leaves: node 3 and node 4, so $P(1) = 2$ (in bold).

Note: You are granted that $P(x+2) < x$ for any $x \geq 2$.

Solution: One important fact that helps to build up the solution is the final note. Without it, we should implement depth-first search on the tree in ASM, using only iterations. However, taking the note into consideration, we can design the following algorithm:

1. loop through every element of the list
2. for every element, mark its father with a special character or number, say, -2
3. at the end, loop again through the list and for every block that is left unmarked, increase the number of the leaves in $P(1)$

The actual ASM code to solve this:

```
LOADI 3
MOVE ACC IN1
MOVE IN1 ACC
LOADIN 1 0
MOVE ACC IN2
LOADI 0
SUBI 2
STOREIN 2 2
MOVE IN1 ACC
ADDI 1
MOVE ACC IN1
SUBI 2
SUB 0
JUMP(<) −10
LOADI 0
STORE 1
LOADI 3
MOVE ACC IN1
LOADIN 1 0
```

```
JUMP(<) 4
LOAD 1
ADDI 1
STORE 1
MOVE IN1 ACC
ADDI 1
MOVE ACC IN1
SUBI 2
SUB 0
JUMP(<) −10
STOP 0
```

Problem 2.2 (Fun Arithmetic) 50pt 10min

Consider the extension of the Boolean Logic in which the universe is $\{0, /, 1\}$ instead of the usual universe $\{0, 1\}$. The operations AND and NOT are defined by the following truth tables:

AND	0	/	1
0	/	1	0
/	1	0	/
1	0	/	1

NOT	
0	/
/	1
1	0

Furthermore, we denote by E^n the expression $E^n = E \wedge E \wedge \ldots \wedge E$. Your task is to design a L(VM) implementation, using all the necessary static procedures, that computes the following expression:

$$(x \wedge \neg y)^n$$

where n is given in stack[0] and x and y are given in stack[1] and stack[2] respectively. You should output the result in stack[3].

Solution:

The solution consists in designing three static procedures for the operations involved in computing the expression, NOT, AND and E^n, where the last procedure is the exponential function, adapted for this situation. The implementation of these procedures is straightforward; for AND, the students can either observe that the result of the computation is a function of the second argument or they can use comparisons again.

```
<NOT>   proc 1 (length)
        con 0 arg 1 leq cjp 5
        con / return
        con / arg 1 leq cjp 5
        con 1 return
        con 0 return
<AND>   proc 2 (length)
        con 0 arg 1 leq cjp 5
        arg 2 call <NOT> return
        con / arg 1 leq cjp 7
        arg 2 call <NOT> call <NOT> return
        arg 2 return
```

```
    <EXP>   proc 2 (length)
            arg 2 con 1 leq cjp 5
            arg 1 return
            con 1 arg 2 sub arg 1 call <EXP>
            arg 1 call <AND> return

    ;final computation
peek 0
peek 1 peek 2
call <NOT> call <AND>
call <EXP>
```

Problem 2.3 (TM for logarithm) 50pt 10min

The logarithm of a number $n \geq 1$, denoted by $\log_2(n)$, is the maximum natural number p such that $2^p \leq n$. You are required to describe a Turing Machine that receives a string of n characters '1' surrounded by a hash at each end, and halts after outputting a string of $\log_2(n) + 1$ characters '1' after the last hash. The tape is right-infinite and filled with special characters (blanks).

Example run:

initial	#1111111111111111#
final	#1111111111111111#11111

You are allowed to use any alphabet. You are **not required** to write the transition table for each state, but **shortly describe** what is the purpose of each state you use.

Note: Your explanation should be clear enough and cover all the cases that might arise. In some situations, writing the transition table might be more clear than using plain English.

Solution: The general idea is to follow the following steps:

1. append a 1 after the last hash
2. proceed to divide the 1's between hashes in two by marking one 1 from the beginning with 0 and a matching 1 from the end with 2
3. revert all 0's back to 1's, go to the beginning of the tape and start again

The clearest solution is a table with explanations:

1	#	1	#	>
1	1	2	1	>
2	1	2	1	>
2	#	3	#	>
3		4	1	<
4	1	4	1	<
4	#	5	#	<
5	2	5	2	<
5	1	6	2	<
6	1	6	1	<
6	#	7	#	>
6	0	7	0	>
7	1	8	0	>
8	1	8	1	>
8	2	5	2	<
5	0	5	1	<
5	#	1	#	>
2	2	2	2	>
3	1	3	1	>
7	2	5	2	<

3 Problem Solving and Search

8pt
15min

Problem 3.1 (Booking the Media Room)

As you are familiar, for booking a Media Room in your favorite college, you need to fill an on-line booking form which ensures that the media room is available for you at the respective time.

You are the web master of the college and you notice a bug in the system: in tomorrow's schedule, a number of bookings were made without ensuring no overlapping. To solve the time-clashes, your task is to validate some of the bookings and cancel the others, while making a maximum number of people happy (validating as many bookings as possible).

Task:

- Represent this as a formal problem (states, start, goal, optimal solution).
- Describe what approach (or algorithm) you would use in order to achieve an optimal solution.
- Exemplify your formulation and your algorithm on the following booking example:

John	11:00 AM - 18:00 PM
Jane	14:00 PM - 15:10 PM
Greg	15:00 PM - 15:15 PM
David	12:00 PM - 13:45 PM
Ann	20:00 PM - 22:30 PM

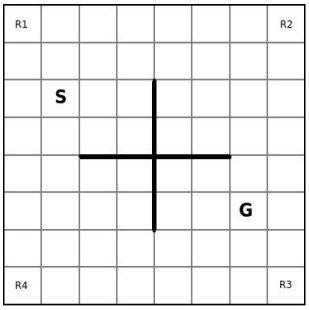
Solution:

- One natural solution to represent this would be to have states which hold the validated bookings. In this manner, the start state is an empty set and with each step, we choose a booking which does not overlap with the ones already validated and place it in the set. A goal state can be defined as a state which does not allow adding new valid bookings. The solution represented by such a goal state will be optimal iff the number of bookings it contains is maximum.
- The optimal approach would be to use a Greedy search and always go to the state which validates a booking with the lowest finish time. This can be proved to be optimal by comparing it with any other solution and noticing that for every valid booking in the new solution has an equivalent valid booking (with an earlier or equal finish time) in the optimal solution.
- Using the algorithm described above, we'll start with no valid bookings and will proceed by choosing the earliest finish time:

{}	
{David}	added David with finish time 13:45
{David, Jane}	added Jane with finish time 15:10
{David, Jane, Ann}	added Ann with finish time 22:30

Problem 3.2 (Robots) 7pt 15min

You are given an 8×8 grid representing the world populated by a person and 4 robots. The person starts in the cell marked with S and wants to escape from the robots. If he reaches cell G (goal), he will be safe. However, there is an obstacle because the bolded borders in the middle of the grid represent walls, thorugh which the person cannot go.



1. First, imagine that the robots are stationary and do not interfere with the motion of the person.

   - The allowed movements are Left, Right, Up and Down, and they allow the person to move from one cell to the adjacent one. The movements have uniform cost. Suggest two admissible heuristics for reaching cell G from cell S. State the estimate of each heuristic for the cost from G to C.
   - Which of the heuristics you suggested dominates the other one?
   - Would your heusristics be admissible if diagonal moves were also allowed (i.e. a move to any of the 8 neighbouring cells is valid)? State why.
   - What is the main difference between A* and Greedy Search? Which one would you choose?

2. Now, imagine that the walls are not there any more, but the robots are in motion. Each of the four robots can go from its initial cell (the ones marked R1, R2, R3, R4) on a diagonal path to the middle of the grid, and back, i.e. each one has 4 cells on which it can step. Each robot can only go from the initial corner cell to the middle, or back, but cannot stop earlier and reverse direction. The person can still move only Left, Right, Up, or Down. In one time step, the person makes one move, and each of the robots makes one move. Suggest a strategy for the movement of the person so that he reaches from S to G without ending up in the same cell with any of the robots on his way.

Solution:

1. For the first part of the problem:

- The real cost is $h* = 10$
  Admissible heuristics are:
  Manhattan distance: 8;
  straight-line distance: $\sqrt{5^2 + 3^2} = \sqrt{25 + 9} = \sqrt{34}$;
  2 x horizontal distance between S and G: $2 \times 5 = 10$;
  2 x vertical distance between S and G: $2 \times 3 = 6$;
  or any other suggestion which makes an estimate between 0 and 10.
- The heuristic that dominates is the one with higher estimate. The above-listed heuristics in order of dominance are: 2 x horizontal distance; Manhattan distance; 2 x vertical distance; straight-line distance.
- The real cost would then be $h* = 6$
  Therefore the Manhattan distance and the 2 x horizontal distance would not be admissible.
  The straight-line distance and the 2 x vertical distance would still be admissible.
- The choice of nodes to expand in Greedy Search is based only on a heuristic function $h(n)$, which estimates the cost from a node to teh goal state. In A* search there is an evaluation function $f(n) = h(n) + g(n)$, which also adds to the estimate the cost from the start to the current state.

2. For the second part of the problem, the person has to escape from the robots at any time step, so we have to watch out for their movement as well. This is easy, because it is clearly defined and we can locate any of the robots at any time.
   Let us number the rows and the columns of the grid 0 thorugh 7, starting from the top left corner, where R1 is located.
   Thus a suggested route is (2,1), (3,1), (3,2), (4,2), (4,3), (5,3), (5,4), (5,5), (5,6).
   Of course, any other path that escapes the robots is accepted.

---

4 Prolog

5pt
9min

Problem 4.1 (The Goldbach Conjecture)

The Goldbach Conjecture is a famous number theoretical problem. The statement is that every even integer greater than 2 can be expressed as the sum of two primes. Write a **ProLog** predicate *prove*(N) which could be used to prove that the proposition holds up to the value N, a given positive integer. It is known that for relatively small values of N ($\leq 10^5$) the statement holds. Your task is to check this for $N = 1000$. In other words, prove that every even integer up to $N = 1000$ can be written as a sum of two primes.

Solution:

```
is_prime(2).
is_prime(3).
is_prime(P) :- integer(P), P > 3, P mod 2 =\= 0, \+ has_factor(P,3).

has_factor(N,L) :- N mod L =:= 0.
has_factor(N,L) :- L * L < N, L2 is L + 2, has_factor(N,L2).

count([], 0).
count([H|T], C) :- count(T, S), C is S + 1.

goldbach(4,[2,2]) :- !.
goldbach(N,L) :- N mod 2 =:= 0, N > 4, goldbach(N,L,3).

goldbach(N,[P,Q],P) :- Q is N - P, is_prime(Q), !.
goldbach(N,L,P) :- P < N, next_prime(P,P1), goldbach(N,L,P1).

next_prime(P,A) :- A is P + 2, is_prime(A).
next_prime(P,A) :- P2 is P + 2, next_prime(P2,A).

prove(4) :- goldbach(4,[2,2]).
prove(N) :- N>4, between(4,N,Y), Y mod 2 =:=0, goldbach(Y,L), count(L,Num), Num =:= 2.
```

---

5 Internet

5pt

Problem 5.1 (Sending emails) 10min

The GenCS TAs were very satisfied by the interest students had for their tutorials and wanted to write a thank you email to everyone. Since the *Information and Software Architecture of the Internet and WWW* topic was not that detailed when they had the class, they are very interested to learn more from you.

1. (briefly) describe the TCP/IP layers that their email passes as it is sent and what transformations happen at each stage.
2. what is HTML and CSS and briefly explain how they can be used to write a *fancy* email (It is sufficient to just write an example email using them both).
3. describe how an asymmetric-key encription method would enable sending confidential messages.

Solution:

1. The layers of the TCP/IP suite, with descriptions, are:

Layer	Description
Application Layer	high level data
Transport Layer	UDP,TCP header added to the data as it gets encapsulated in a packet
Internet Layer	IP header and data
Link Layer	packets are divided to frames

2. HTML is a representation format for web pages and CSS is a style sheet language that allows authors and users to attach style (e.g., fonts and spacing) to structured documents. A basic example with HTML tags and CSS style is sufficient to show the use of them.
3. In an asymmetric-key encryption method, the key needed to encrypt a message is different from the key for decryption. Such a method is called a public-key encryption if the encryption key (called the public key) is very diffcult to reconstruct from the decryption key (the private key). To send a confidential message the sender encrypts it using the intended recipient's public key; to decrypt the message, the recipient uses the private key.
