NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

EXAMINATION FOR
Semester 2 AY 2013–2014

GEM1501 – PROBLEM SOLVING FOR COMPUTING

April/May 2014 Time Allowed 2 Hours

INSTRUCTIONS TO CANDIDATES

1. This examination paper consists of TEN (10) questions and comprises ELEVEN (11) printed pages, including this page.
2. Answer **ALL** questions within the space in this booklet.
3. This is a **Closed Book** examination.
4. Please write your Matriculation Number below:

   MATRICULATION NO: _____

5. You may use calculators, provided that they do not contain any program or memory content.
6. Every question carries FIVE (5) marks and there are FIFTY (50) marks in total.

This portion is for examiner's use only

Qestion	Marks	Remarks	Qestion	Marks	Remarks
Q01:			Q06:		
Q02:			Q07:		
Q03:			Q08:		
Q04:			Q09:		
Q05:			Q10:		
			Total:		

Question 1 [5 marks] **GEM 1501**

Explain what a punch card is and for which early machines (before 1900) it was used.

Solutions. A punch card is rectangular carton card which is used to store information. Punch cards can store patterns for weaving, information for census or computer programs. The earliest usage was to control looms by punch cards which represented the patterns in the textiles to be weaved. Charles Babbage and later entrepreneurs wanted to use punch cards to feed their calculating machines with computer programs and inputs. The punch card had it breakthrough in information processing when Herman Hollerith utilised it to organise the 1890 census in the USA: The census data was punched into cards and then tabulating machines computed from stacks of such cards the statistical data required by the census bureau.

Question 2 [5 marks] **GEM 1501**

The following JavaScript program contains syntax and other programming errors in five of its lines. List them out below and explain what is wrong.

```
function calc(x)      // Line 1
  { array a; integer y,z; // Line 2
    z = calc(x);            // Line 3
    for (y=0;y<55555)          // Line 4
      { z = z+y; a[y] = z;        // Line 5
        switch(z)                    // Line 6
          { case 22: z = z+8; break;    // Line 7
            case 24: z = z+y; break;       // Line 8
            case default: z = 22; break; } }  // Line 9
     if (z < 3000) then { return(a); }           // Line 10
     a[0] = z; return(a); }                         // Line 11
```

Solutions.

- Line 2: In JavaScript, variables are just declared with the word "var" and types are not used in the declaration. Types are linked to the value and not to the variable which holds the value.
- Line 3: This recursive call of calc redirects to the beginning of the function without a terminating base case and causes an infinite loop.
- Line 4: The usual for-statement has three components, one for counting up the variable `y` is missing.
- Line 9: The word `default` does not have a `case` in front of it; it should just be omitted.
- Line 10: The word `then` is not utilised in if-then-else statements in JavaScript. Although it would make sense to use the word here, it actually causes a syntax error.

Question 3 [5 marks] **GEM 1501**

Consider the following JavaScript program.

```
function f(a)
  { var n = a.length;
    var i; var j;
    var b = new Array(n);
    for (i=0;i<n;i++)
      { b[i] = 0;
        for (j=i;j<n;j++)
          { b[i] += a[j]; } }
    return(b); }
```

What is the time complexity of this program?

☐ $\Theta(n)$ ☐ $\Theta(n \cdot \log(n))$ ☐ $\Theta(n^2)$ ☐ $\Theta(n^2 \cdot \log(n))$

This program can be improved significantly. What is the time complexity of your improved program (it should be better than the above)?

☐ $\Theta(\log(n))$ ☐ $\Theta(n)$ ☐ $\Theta(n \cdot \log(n))$ ☐ $\Theta(n^2)$

Write your program below.

Solutions. The time complexity of the program is $\Theta(n^2)$. The program can be improved to time complexity $\Theta(n)$ by using the formula that `b[i] = a[i]+b[i+1]`. The corresponding program is the following:

```
function f(a)
  { var n = a.length;
    var i; var s = 0;
    var b = new Array(n);
    for (i=n-1;i>=0;i--)
      { s = s+a[i];
        b[i] = s; }
    return(b); }
```

Question 4 [5 marks] **GEM 1501**

Let $a \oplus b \oplus c$ be 1 if and only if an odd number of the input variables a, b, c have the value 1 (and similarly for other numbers of inputs). Furthermore, let $a \wedge b$ be 1 if and only if both inputs a and b are 1. Now consider

$$F(a, b, c, d) = (a \wedge b) \oplus (a \wedge c) \oplus (a \wedge d) \oplus (b \wedge c) \oplus (b \wedge d) \oplus (c \wedge d).$$

Describe in words when $F(a, b, c, d)$ is 1 and when it is 0. Furthermore, how many of the binary vectors (a, b, c, d) are mapped to 1?

Solutions. In the definition of F, all possible conjunctions of two input-variables occur and the output is 1 if and only if an even number of such conjunctions is 1. Thus the output value only depends on the number of input variables which are 1.

If zero or one inputs are 1 then the output is 0; if two inputs are 1 then exactly one conjunction of two inputs is 1 and so the output is 1; if three inputs are 1 then three conjunctions are 1 and the output is 1; if all inputs are 1 then an even number of conjunctions are 1 and the output is 0. In summary function F takes the value 0 if zero, one or four inputs are 1 and takes the value 1 if two or three inputs are 1.

There are six binary four-tuples with two 1s and four binary four-tuples with three 1s, hence in total ten input-vectors are mapped to 1.

Question 5 [5 marks] **GEM 1501**

Describe the following three machines / automata and point out the differences between them:

- a finite automaton;
- a linear bounded Turing machine (also known as linear bounded automaton);
- a Turing machine (without resource bound constraints).

Solutions. A Turing machine has a finite number of states and works on a tape which is initially filled up with finitely many non-blank symbols and before and after the input are an infinite amount of blank symbols on the tape. The head of the Turing machine stands initially on the first non-blank symbol and the Turing machine is in the starting state. In each cycle, the Turing machine reads the current symbol and choses according to its Turing table an activity which could be replacing the symbol by another symbol or going one field to the left or going one field to the right. Furthermore, it chooses according to the table a new state. The Turing machine halts when it reaches a halting state. The output of the Turing machine is the content of the tape at the time of halting, starting from the first non-blank symbol up to the position before the first blank symbol behind it.

A linear bounded automaton is like a Turing machine with the only difference that it does not overwrite any blanks; that is, the blanks left and right of the input word will not be changed. Normally, one is only interested in linear bounded automata which accept or reject words; therefore one can introduce two types of halting states, one for accepting and one for rejecting the input.

A finite automaton is in principle a Turing machine which reads the current input symbol, changes the state and goes right; it does never modify the tape. It has accepting and rejecting states and it halts when it reaches the first blank after the input; the current state then decides whether the machine rejects or accepts the input.

Linear bounded automata can recognise much more languages than finite automata and Turing machines can recognise much more languages than linear bounded automata. However, one can easily modify finite automata and check their properties, for example whether they accept any word. Such tests are impossible for the other types of machines.

Question 6 [5 marks] **GEM 1501**

The Roman general and politician Gaius Julius Caesar used a cryptographic method to encrypt and decrypt texts. Explain the method and also its vulnerabilities: In particular, utilise the knowledge that "L" is a very frequent letter to decrypt the message "DAHHK SKNHZ!"; give the full value of the message; note that spaces and punctuation marks are not encrypted.

Solutions. The method of cryptography used by Ceasar was to define an offset and to exchange each letter in the text by a letter which is by the offset more behind in the alphabet; the last letters are then accordingly mapped to the front of the alphabet. This type of cryptography has two vulnerabilities. First, there are only 25 possible keys, representing an offset 1 up to an offset 25. Second, in long texts, the codes of the most frequent letters occur most frequently in the encryption, giving away what these codes stand for. In the current encoding, "H" stands for "L" which gives an offset of 22. So for decryption, one has to replace every letter by the letter which is 4 positions behind and one receives the message "HELLO WORLD!" as the plain text.

Question 7 [5 marks] **GEM 1501**

Write a function `sort(a)` with the following properties:

- The input `a` is a dynamical array (with some indices used and others not);
- The output `b` is an empty array at the beginning and should contain the elements of `a` in sorted order after the end of the function.

For example, if `a[5]` is 12, `a[8]` is 11 and `a[17]` is 22 and no other elements of `a` exist, then the function should define the array elements `b[0]` to be 11, `b[1]` to be 12 and `b[2]` to be 22.

Solution.

```
function sort(a)
  { b = new Array();

    var i; var j;
    for (i in a)
      { j = b.length;
        while ((j>0)&&(b[j-1]>a[i]))
          { b[j] = b[j-1]; j--; }
        b[j] = a[i]; }

    return(b); }
```

Question 8 [5 marks] **GEM 1501**

Make a deterministic finite automaton using the alphabet $\Sigma = \{0, 1, 2\}$ such that the automaton accepts a word w if and only if the word contains exactly two times a 1 and exactly one time a 2.

Solution. The automaton has the states (a, b) with $0 \leq a \leq 2$ and $0 \leq b \leq 1$. If the automaton reads a 0, it goes from the state (a, b) to (a, b); if it reads a 1 and $a < 2$ it goes from the state (a, b) to the state $(a + 1, b)$; if it reads a 2 and $b < 1$ it goes from the state (a, b) to $(a, b + 1)$. In all other cases, the automaton gets stuck. The state $(2, 1)$ is the unique accepting state and the state $(0, 0)$ is the starting state.

Question 9 [5 marks] **GEM 1501**

The following regular expression

$$30^* \mid 20^*10^* \mid 10^*20^* \mid 10^*10^*10^*$$

describes a set of decimal numbers. Give a description of this set in words and list all elements of length one or two. For example, 10^* describes the set of all powers of 10 and the elements up to length five are 1 (one), 10 (ten), 100 (hundred), 1000 (one thousand) and 10000 (ten thousand).

Solutions. The set described is the set of all numbers with are the sum of three powers of ten, for example 210 is in the set and is equal to hundred plus hundred plus ten. The elements of length one or two are 12, 21, 3 and 30. Note that 1 is the zeroth power of 10 in this context and can be used to form the sum of three powers of ten.

Question 10 [5 marks] **GEM 1501**

A solution of a set of clauses is a truth-assignment to the logical variables such that each clause is satisfied by the given assignment. The satisfiability problem asks whether for a given set of clauses there is at least one truth-assignment making all clauses true. Consider the following set of clauses:

- $x_1 \vee x_2 \vee x_3 \vee x_5$;
- $\neg x_1 \vee x_4$;
- $\neg x_2 \vee x_4$;
- $\neg x_3 \vee x_4$;
- $\neg x_4 \vee \neg x_5$;
- $x_5 \vee x_6 \vee x_7$;
- $x_5 \vee \neg x_6$;
- $x_5 \vee \neg x_7$.

How many solutions does this set of clauses have?

☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 or more.

Write a few lines on how you determined the number of solutions.

Solutions. The last three clauses can only be true at the same time in the case that x_5 is true; however, when x_5 is true, then x_4 is false, x_1, x_2, x_3 are false and there are no constraints on the values taken by x_6 and x_7. Hence there are 4 solutions to the set of clauses.

END OF PAPER