NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

SEMESTER II: 2010–2011

EXAMINATION FOR

GEM 1501 – Problem Solving for Computing

April 2011 – Time Allowed 2 Hours

INSTRUCTIONS TO CANDIDATES

1. This examination paper consists of TEN (10) questions and comprises ELEVEN (11) printed pages.
2. Answer **ALL** questions.
3. This is a **Closed Book** examination.
4. Every question counts FIVE (5) marks. The maximum possible marks are 50.
5. Please write your Matriculation Number below:

SOLUTIONS

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**

What is an array and what is a record? Describe the two data types and explain the differences between them. How is a record realised in Java Script?

Solution. Arrays and records are two ways to combine various data-items into one data-structure. In an array, say with name ar, one usually has similar data-items which are just indexed as $ar[0], ar[1], \ldots, ar[n-1]$ where n is the length of the array. In a record, there are only a few data-items which are distinguished by fixed names, say $re.name$, $re.homephone$ and $re.handphone$ for a record re having the three data-items of a name plus two phone numbers. In Java Script, such entries are made using constructor functions which are then used together with the keyword "New" in order to create a record with the corresponding entries; one can then access the entries by giving the variable name followed by a dot and the component name. Here an example.

```
function remake(na,ho,ha)
  { this.name = na;
    this.homephone = ho;
    this.handphone = ha; }
re = New remake("Andrew","66665555","88889999");
document.write("The person is "+re.name);
document.write(" and his number is "+re.handphone+".<br>");
```

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

There are places $a_0, a_1, a_2, \ldots, a_{12}$. Assume that the direct fare for going from a_i to a_j (with $i < j$) is $8 + (j - i) \cdot (j - 1 - i)/2$. Find the cheapest route to go from a_0 to a_{12} consisting of several direct steps using dynamic programming. Determine the price and list one possible route.

Solution. First one makes a table what it costs to travel the distance from a_i to a_j; this only depends on the difference $j - i$:

distance	1	2	3	4	5	6	7	8	9	10	11	12
price	8	9	11	14	18	23	29	36	44	53	63	74

Now one uses this table in order to compute for $n = 11, 10, 9, \ldots, 0$ the price and the next node of a route.

Node	Next Node	Price	Alternative	Price
a_{11}	a_{12}	8	—	—
a_{10}	a_{12}	9	a_{11}	16
a_9	a_{12}	11	a_{11}	17
a_8	a_{12}	14	a_{10}	18
a_7	a_{12}	18	a_{10}	20
a_6	a_9	22	a_{12}	23
a_5	a_9	25	a_8	25
a_4	a_8	28	a_7	29
a_3	a_8	32	a_7	32
a_2	a_7	36	a_6	36
a_1	a_4	39	a_5	39
a_0	a_4	42	a_5	43

So the total price is 42 and the route is $a_0 \to a_4 \to a_8 \to a_{12}$.

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

Consider the following function `ijksearch` which has as input an array x of length n. The array members are numbers.

```
function ijksearch(x)
  { var n = x.length;
    var i; var j; var k;
    for (i=0;i<n;i++)
      { for (j=i+1;j<n;j++)
          { for (k=j+1;k<n;k++)
              { if ((x[i]==x[j])&&(x[i]==x[k])) { return(1); } } } }
    return(0); }
```

What is the worst-case complexity of the running-time of this function? Mark the opitmal answer:

☐ $O(1)$ ☐ $O(\log(n))$ ☐ $O(\log^2(n))$ ☐ $O(n\log(n))$ ☐ $O(n^2)$
☐ $O(n^2\log(n))$ ☒ $O(n^3)$ ☐ $O(n^3\log(n))$ ☐ $O(n^4)$ ☐ $O(n^5)$

What is the worst-case complexity of the problem, that is, the worst-case complexity of the fastest program having the same input-output behaviour? The program is permitted to make copies of the array and compare the array elements; copying each array element and comparing two array elements always induces the cost of one time-unit. Furthermore, all standard comparisons $<$, $<=$, $!=$, $==$, $>$, $>=$ are permitted. Mark the optimal answer describing the complexity of the problem:

☐ $O(1)$ ☐ $O(\log(n))$ ☐ $O(\log^2(n))$ ☒ $O(n\log(n))$ ☐ $O(n^2)$
☐ $O(n^2\log(n))$ ☐ $O(n^3)$ ☐ $O(n^3\log(n))$ ☐ $O(n^4)$ ☐ $O(n^5)$

Give reasons for your answers.

Solution. The function `ijksearch` determines whether there are distinct i, j, k where the array x has the same entries. This is done with three nested loops searching over all ordered triples (i, j, k); there are $\frac{n(n-1)(n-2)}{6}$ many of these triples giving the corresponding cubic bound.

One can simplify the task by sorting the data in $O(n\log(n))$ with an algorithm like mergesort and then check whether there is an entry $m < n-1$ such that in the sorted copy y it holds that $y[m]$, $y[m+1]$ and $y[m+2]$ are all the same string. As the strings are linearly ordered and merge sort can handle duplicates, this algorithm delivers the required time bounds. As the data in x is not ordered originally, every element has to be touched at least once and sublinear timebounds do not apply.

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

What are 2SAT and 3SAT? Furthermore, how does the algorithm "Resolution" behave on these two problems? What difference is observed and what is the reason for it?

Solution. A 2SAT instance is a list of clauses of the form x_i, $\neg x_i$, $x_i \vee x_j$, $x_i \vee \neg x_j$, $\neg x_i \vee x_j$ and $\neg x_i \vee \neg x_j$ over some truth-variables $x_1, \ldots, x_n$ where n denotes the size of the input. A 2SAT instance is satisfiable when one can assign the truth values to the variables in a way that every clause is evaluated to true. Similarly for 3SAT with the only difference that clauses can have up to three variables.

Resolution can handle 2SAT in polynomial time while it needs exponential time for some 3SAT instances. The reason is that in the resolving step, in the case of 2SAT the new clause is again only containing two literals while for 3SAT, the subsequent resolving steps make the new clauses longer and longer and their number more and more numerous. This gives some evidence that NP-complete problems like 3SAT have indeed a higher runtime complexity than problems in P like 2SAT, although this is by now not yet proven.

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

Write a counter program which computes on input n the number a_n with $a_0 = 1$, $a_1 = 2$ and $a_{n+2} = a_n \cdot a_{n+1}$ for all n. For this task, the counter program is permitted to use operations to compare, to add and to subtract, along with control operations while, for and if. It is permitted to write subfunctions and to call them. All numbers $0, 1, 2, \ldots$ can be used as constants. However, multiplication and division operations are not available. The input n is always a natural number.

Solution. This can be solved with a function which first computes the corresponding Fibonacci number and then takes the power of 2 of it.

```
function cprg(n)
  { var i; var j; var m; var k;
    i=0; j=1; k=0;
    if (n==0)
      { k = 0; }
    else if (n<3)
      { k = 1; }
    else
      { i=1; j=1;
        for (m=2; m<n; m=m+1)
          { k = i+j; i=j; j=k; } }
    m = 1;
    while (k>0)
      { m = m+m; k = k-1; }
    return(m); }
```

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

Make a finite automaton (as a graph or a table) which accepts inputs consisting of binary numbers and operands "+", "-" and "∗". Between any two operands and before the first operand and after the last operand should be a number. Also, except for 0, all binary numbers start with "1", so there are no leading zeroes.

Here some examples. The automaton should accept the following inputs:
`0+11+10*11-101+111*11`
`110101010001-101010010101010`
`1+0+1+0*0-1`
It should reject the following inputs:
`00+10+00101+101+10+110`
`+1+0+11+0+1`
`1**0*1-1101+11`

Solution. The finite automaton has the following table where names refer to the currently processed syntactic unit and "error" is an error state not left on any input.

name of state	0	1	+	−	∗	acc/rej
start	null	number	error	error	error	reject
null	error	error	operand	operand	operand	accept
number	number	number	operand	operand	operand	accept
operand	null	number	error	error	error	reject
error	error	error	error	error	error	reject

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

Caesar's cryptography algorithm permutated the letters so that "a" became "d", "b" became "e", "c" became "f", ..., "z" became "c". Write a Java Script function which permutates the (lower case) letters in this way and lets all other symbols unchanged. Input and output are strings x and y. So "the river is yellow." should be mapped to "wkh ulyhu lv bhoorz." by the following function "ceasar".

Solution. The following program searches for each symbol of x in the list of letters whether it appears and puts into y the coded copy. Punctuation marks and spaces remain unchanged.

```
function caesar(x)
  { var y = "";
    var letters = "abcdefghijklmnopqrstuvwxyz";
    var n = x.length;
    var m; var k; var c; var d;
    for (m=0;m<n;m++)
      { c = x.charAt(m); d = c;
        for (k=0; k<letters.length;k++)
          { if (c == letters.charAt(k))
              { d = letters.charAt((k+3)%26); } }
        y = y+d; }
    return(y); }
```

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

Explain what is Nick's Class.

Given n and n numbers $a_1, a_2, \ldots, a_n \in \{0, 1, 2, 3, 4\}$ the task is to compute the remainder of $a_1 \cdot a_2 \cdot \ldots \cdot a_n$ when divided by 5. For example, if $n = 6$ then the output for $(3, 0, 2, 3, 4, 3)$ is 0 and the output for $(1, 1, 2, 1, 4, 1)$ is 3.

Is this problem in NC (Nick's Class)? [x] Yes, [ ] No.

Explain why the problem is inside or outside Nick's class.

Solution. Nick's class consists of problems which can be solved by a network of gates with constantly many ingoing wires from lower levels and constantly many outgoing wires to higher levels such that the number of levels is polylogarithmic in the length n of the input and the number of gates is polynomially in n (or less).

The given problem can be shown to be inside NC by a divide and conquer algorithm which combines always on each level two neighbouring outputs from the previous level into one new output for the upper level which is their product. It looks like the maximum-circuit from the lecture, only that it uses here multiplication in a finite field of characteristic 5.

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

Describe what the Turing test is. Explain the conditions which a computer has to satisfy to pass the test. What are the challenges which one has to handle when making such a computer program.

Solution. The Turing test is a test whether a machine can be so intelligent that a human cannot distinguish it from a fellow human. It consists of a human evaluator who talks about various chat channels with either machines or other humans. The evaluator wins if he can identify those chats which are with machines and those which are with other humans; the machine passes the Turing test if the majority of human evaluators fail to do so.

There are various challenges to pass: First, a machine running successfully in the environment needs good skills in language processing, needs to understand the content of the language and be familiar with the basic facts which the humans in the control group might know. Second, it should not be too intelligent and copy dozens of Wikipedia pages to answer a question as no human would give such a comprehensive and perfect answer to a question. Third, jokes and other strange behaviour of humans should be properly dealt with.

In short, the machine should behave like an average human and have all the faults and strengths of a human without exagerating them.

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

Write a function `count(x)` which counts for input x how many factors p of x exist with the following properties: p is a prime number, $p+2$ is a prime number, $p(p+2)$ is a factor of x. So `count(630)` is 2 as witnessed by $p = 3$ (here 3 is a prime, 5 is a prime and $3 \cdot 5$ divides 630) and $p = 5$ (here 5 is a prime and 7 is a prime and $5 \cdot 7$ divides 630).

```
function isprime(y)
  { if (y<2) { return(0); }
    if (y<4) { return(1); }
    int z;
    for (z=2;z<y;z++)
      { if (y%z==0) { return(0); } }
    return(1); }

function count(x)
  { var p; var c = 0;
    for (p=3;p<x-3;p++)
      { if (isprime(p)&&isprime(p+2)&&(x%p==0)&&(x%(p+2)==0))
          { c++; } }
    return(c); }
```