NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

SEMESTER II: 2007–2008

EXAMINATION FOR

GEM 1501 – Problem Solving for Computing

Monday 02 May 2008 Morning – Time Allowed 2 Hours

INSTRUCTIONS TO CANDIDATES

1. This examination paper consists of TEN (10) questions and comprises TWELF (12) printed pages.
2. Answer **ALL** questions.
3. This is an **Closed Book** examination.
4. Every question counts FIVE (5) Marks which is distributed equally on sub-questions in the case that there are any. The maximum possible marks are 50.
5. Please write your Matriculation Number below:

MATRICULATION NO: ________________________________

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**

The following people contributed to the early history of algorithmic and computing: Charles Babbage, Diophantus, Euclid, Mohammed al-Khowârizmî, Ada Lovelace, Herman Hollerith, John von Neumann, Blaise Pascal, Alan Turing and Konrad Zuse. Answer the following questions to the history of computing.

(a) In his text-book on geometry, __Euclid__ formulated the first non-trivial algorithm in order to compute the greatest commond divisor of two integers. Some centuries later, __Diophantus__ created a mathematical theory of polynomial equations like $x^2 + y^3 = 22$ and studied their solutions; these type of equations and sets are named after him.

(b) In the nineth century, __Mohammed al-Khowârizmî__ created the foundations of school algebra as we know it today. He formulated many algorithms to add, multiply and divide numbers. He also developed methods to solve linear and quadratic equations. The word "algorithm" originates from his surname.

(c) The age of mechanical computers predates modern computing by many centuries. The French mathematician __Blaise Pascal__ is well-known for his mechanical calculator which could do basic arithmetic operations with numbers. The British scientist __Charles Babbage__ attempted to build a fully programmable computer on a mechanical basis, but his machine got never ready. His assistant __Ada Lovelace__ aided him by developping computer programs for this machine although it never became ready.

(d) At the end of the nineteenth century, electronic equipment became more and more pupular for computing machinery. The American __Herman Hollerith__ developed a whole machinery to run and evaluate censuses. He won a competition to code and evaluate the census of 1890 in the USA and his machinery brought down the processing time from 7 years for the last census to six weeks with a method based on punch cards.

(e) In the years 1940 to 1950 the first fully programmable computers where developed. Scientists worked on it in various countries. The German __Konrad Zuse__ developed a whole series and was able to run a company for the first years after the end of the war. The British mathematician __Alan Turing__ developed the theoretical foundations for computers and was also involved in a project to crack the military codes of the Axis countries during World War II. Since then, cryptography is one major applications of computing. In America, the Hungarian __John von Neumann__ developed a basic architecture for computers; this architecture became named after him and is until today used by many manufacturers.

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

Consider the following list of problems. Put them into the corresponding category. Here n is the size parameter of a problem. Here the descriptions:

- Halting Problem: Given a program P and an input I, does P with input I halt and produce some output?
- Prime: Given an n-digit decimal number m, is m a prime number?
- 2SAT: Given a list of conditions of the form $x_1 \vee x_2$, $\neg x_3 \vee x_4$, $\neg x_5 \vee \neg x_6$ using at most n variables, does it have a common assignment satisfying all conditions?
- 3SAT: The same as 2SAT, but the conditions can involve three variables like $x_1 \vee x_2 \vee \neg x_3$.
- Bounded Monkey Puzzles: Given a set of tiles and a fixed area of size $n \times n$, can this area be covered with tiles from the set?
- Diophantine Equation: Given a polynomial p with integer coefficients and a parameter x, are there natural numbers $y_1, y_2, ..., y_n$ with $p(x, y_1, y_2, ..., y_n) = 0$?
- Unbounded Monkey Puzzles: Given a set of tiles, can it cover an $n \times n$ area (with repetitions of tiles) for every n?
- Totalness Problem: Given a program P, does it halt on every possible input I?
- Checkers: Given a situation of checkers on an $n \times n$ board, can white win the game from this situation when white and black play both as good as possible?
- Presburger Arithmetic: Given a formula using integer numbers, $<$ and addition, is this formula true? Examples of such formulas are $\forall x \exists y\, [x + x + x = y + y]$ and $\exists x \forall y\, [x \neq y + y + y + y + y]$.

Find for each of the following case two of the above problems that fit.

(a) Problems known to be solvable in polynomial time:
2SAT, Prime.

(b) Problems known to be NP-complete:
3SAT, Bounded Monkey Puzzle.

(c) Problems harder than NP-problems but known to be decidable:
Checkers, Presburger Arithmetic.

(d) Recursively enumerable but undecidable problems:
Halting Problem, Diophantine Equation.

(e) Problems which are not recursively enumerable:
Totalness Problem, Unbounded Monkey Puzzle.

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

What is the status of the following mathematical statements? The answer can be YES if scientists know that the statement is true, the answer can be NO if scientists know that the answer is false and the answer can be OPEN if scientists do not know the answer and it is an important open problem.

(a) Is P = NP?
☐ YES, ☐ NO, ☒ OPEN.

(b) Is every recursively enumerable set either decidable or complete (for r.e. sets)?
☐ YES, ☒ NO, ☐ OPEN.

(c) Are there problems which can be decided in exponential time but not in polynomial time?
☒ YES, ☐ NO, ☐ OPEN.

(d) Can every decidable language be accepted by a finite automaton?
☐ YES, ☒ NO, ☐ OPEN.

(e) Is Nick's class NC different from P?
☐ YES, ☐ NO, ☒ OPEN.

Assume that f is a strictly monotonic growing function (given as a subprogram), n is a natural number and y a value with $f(0) < y < f(n)$. The goal is to find an integer x with $f(x) \leq y < f(x+1)$ such that the subprogram f is called as seldom as possible. Complete the following information about a strategy called "binary search" or "halving search" to find x.

(a) The initial values are $i = 0$ and $j = n$ and the search interval is of the form $\{i, i+1, i+2, \ldots, j-2, j-1\}$. The program runs a loop such that each time some point k in the interval is taken and $f(k)$ is evaluated and then a corresponding update is done. The loop is run as long as the condition

☐ $i > j$ ☒ $i + 1 < j$ ☐ $f(i) < f(j)$ ☐ $f(i) * f(j) < 1$

is true.

(b) The value k is chosen such that

☐ $k = i + 1$ ☐ $k = j - 1$ ☐ $k = i+\text{Math.floor}(\frac{(j-i)\cdot(f(j)-y)}{f(j)-f(i)})$

☒ $k = \text{Math.floor}(\frac{i+j}{2})$ ☐ $k = i + \text{Math.floor}((\text{j-i})\cdot\text{Math.random}())$.

The update rules it that if $f(k) \leq y$ then $i = k$ else $j = k$. This update rule shrinks the interval in each round and so the algorithm terminates after finitely many rounds.

(c) Let $ncf(n)$ denote the number of calls of f used in the worst case to find x given n. Determine among order of $ncf(n)$ by ticking that term $O(g(n))$ such that $O(g(n)) = O(ncf(n))$:

☐ $O(1)$ ☐ $O(\log\log(n))$ ☒ $O(\log(n))$ ☐ $O(\log(n)\log\log(n))$

☐ $O(n)$ ☐ $O(n\log(n))$ ☐ $O(n^2)$ ☐ $O(n^2\log(n))$ ☐ $O(2^n)$.

(d) If $n = 10000$ which of the following numbers is the nearest to the number of calls of f made by the "halving search" algorithm:

☐ 1 ☐ 3 ☒ 9 ☐ 27 ☐ 81 ☐ 243 ☐ 10000 ☐ 10^{100}.

(e) Is there any algorithm which can find x by $n = 1000$ with only 5 calls to compute $f(k)$ for some values k where the only additional information about f (except these 5 values) available is that $f(0) < y < f(1000)$ and that f is strongly monotonic increasing.

☐ Yes, this binary search algorithm does it;

☐ Yes, some other algorithm does it, but the binary search algorithm does not.

☒ No, no algorithm can do this.

Consider the following finite automaton:

```
List of all states: A,B,C,D;
Starting state: A;
Accepting state: D;

Transition-Table of Automaton:

Old State  |   Input  | New State
-----------+----------+----------
   A       |    0     |   B
   A       |    1     |   A
   B       |    0     |   B
   B       |    1     |   C
   C       |    0     |   C
   C       |    1     |   D
   D       |    0     |   C
   D       |    1     |   D
```

Which of the following words are accepted by the automaton?

(a) 01101 [x] accept; [ ] reject.

(b) 01100 [ ] accept; [x] reject.

(c) 11011 [x] accept; [ ] reject.

(d) 11101 [ ] accept; [x] reject.

(e) Describe in a few words how the language accepted by the automaton looks like.

The automaton accepts words which have 01 as a substring not covering the last digit and where the last digit is also a 1.

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

This question deals with cryptography.

(a) In RSA public cryptography every participant has a private and a public key. The private key are a pair of natural numbers x, y, the public key is a number z. How is z computed from the numbers x and y? Please give the formula here: $\underline{\quad z = x \cdot y \quad}$.

(b) There are two functions, Encrypt_A and Decrypt_A using the keys of person A. Which of the functions uses the private and which uses the public key?

[x] Encrypt_A uses public key z and Decrypt_A uses private key x, y.
[ ] Encrypt_A uses private key x, y and Decrypt_A uses public key z.

Modern usage of the protocol uses on both sides a two-stage coding and decoding; the sender of the message uses the public key of the receipient plus his private key, the receiver uses his private key plus the public key of the sender. Which additional property besides keeping the message secret is the reason for this? — Tick the most appropriate answer only.

[ ] The secrecy of the messages is more protected by double coding.
[ ] It is more difficult to modify intercepted messages.
[x] It is more difficult to send a message under another person's name.
[ ] The method introduces additional redundant bits against noise.

(c) Assume that scientists found out that P = NP. Alice and Bob keep communicating by long messages, each encrypted with one single short mesage encrypted and decrypted with one short key. Alice and Bob know the software with which Charles wants to read their messages he intercepts. Can Alice and Bob choose the key k such that Charles cannot decode the encrypted version E of the next message M which Alice is going to send to Bob?

[ ] Yes, they can. When Alice wants to send a message M to Bob, she can for every key k check whether Charles' program can decipher the encrypted message $Enc(M, k)$ and then select that key k for which Charles fails. The search for this k is fast as due to P = NP a fast search algorithm is known.

[x] No, they can not. Charles can intercept the encrypted message E of Alice sent to Bob and then test any short key k' which might be used by Bob to decrypt the message. If k' is correct, the decrypted message $D(E, k')$ follows the statistical properties of natural language concerning the distribution of words and letters; otherwise this statistical test fails. The search for k' is fast (due to P = NP) and hence Charles gets hold of the original text of the message.

(d) One encryption scheme uses for bit-wise operations with private keys as long as the encrypted message, for every message sent taking a new key. Which operation is it:

- [ ] `E = M & k` (bitwise and),
- [x] `E = M ^ k` (bitwise exclusive or),
- [ ] `E = M | k` (bitwise inclusive or).

(e) One of the oldest methods to encrypt messages was developed by the Roman emperor Cesar which replaced the latters in the alphabet in a systematic way. Do you remember the code? Use it to decrypt the following English language text:

"D fdw fdwfkhv plfh, d grj fkdvhv fdwv."

A cat catches mice, a dog chases cats.

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

Five programmers want for a competition write a program computing the factorial with using additions only. The input n is a natural number. Evaluate the proposed programs as "Okay", "Exponential time" (in the parameter n, not in size of n), "Has syntax-errors" and "Not terminating". A program which needs exponential time is not okay as it can be done in polynomial time.

```
(a) function factorial(n)
      { var k = factorial(n-1); var h = 0; var m = 0;
        while (h<n) { m+=k; h++; } return(m); }
```

☐ Okay; ☐ Exponential time; ☐ Has syntax-errors; ☒ Not terminating.

```
(b) function subfactorial(n,m)
      { if (m<1) { return(0); }
        else if (n<2) { return(1); }
        else return(subfactorial(n,m-1)+subfactorial(n-1,n-1)); }
    function factorial(n) { return(subfactorial(n,n)); }
```

☐ Okay; ☒ Exponential time; ☐ Has syntax-errors; ☐ Not terminating.

```
(c) function factorial(n)
      { var i,j,m,k; m=1; k=0;
        for (i=1;i<=n;i++)
          { for (j=1;j<=i;j++) { k += m; }
            m = k; k=0; }
        return(m); }
```

☒ Okay; ☐ Exponential time; ☐ Has syntax-errors; ☐ Not terminating.

```
(d) function factorial(n)
      { if (n<2) { return(1); }
        var h = factorial(n-1); var k = n; var m = 0;
        while (k>0) { m += h; k--; } return(m); }
```

☒ Okay; ☐ Exponential time; ☐ Has syntax-errors; ☐ Not terminating.

```
(e) function factorial(n)
      { if (n<2) { return(1) }
        var h = factorial(n-1); var k = n; var m = 0
        for (k>0;k--) { m += h } return(m) }
```

☐ Okay; ☐ Exponential time; ☒ Has syntax-errors; ☐ Not terminating.

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

Evaluate the order of the runtime of the following sample programs in terms of the length n of the array processed. Take the runtime of subfunctions into account when computing the runtime of a function; operations with numbers like addition and multiplication have cost 1. Mark the optimal order of the runtime of a function and not a larger one.

(a)
```
function sumn(ar)
   { var n = ar.length; var i,j; var s = 0;
     for (i=0;i<n;i++)
       { for (j=0;j<n;j++)
           { s+=ar[i]*ar[j]; } }
     return(s); }
```

Runtime is ☐ $O(1)$, ☐ $O(\log n)$, ☐ $O(n)$, ☐ $O(n \log n)$, [x] $O(n^2)$, ☐ $O(n^2 \log n)$, ☐ $O(n^3)$, ☐ $O(n^3 \log n)$, ☐ $O(n^4)$, ☐ $O(n^4 \log n)$.

(b)
```
function splitsum(ar)
   { var n = ar.length; var k = Math.floor(n/3);
     var br = ar.slice(0,k); var cr = ar.slice(k,n);
     return(sumn(br)*sumn(cr)); }
```

Runtime is ☐ $O(1)$, ☐ $O(\log n)$, ☐ $O(n)$, ☐ $O(n \log n)$, [x] $O(n^2)$, ☐ $O(n^2 \log n)$, ☐ $O(n^3)$, ☐ $O(n^3 \log n)$, ☐ $O(n^4)$, ☐ $O(n^4 \log n)$.

(c)
```
function position(ar,w)
   { var n = ar.length; ar.push(w); var k = 0;
     while (ar[k]!=w) { k++; }
     return(k); }
```

Runtime is ☐ $O(1)$, ☐ $O(\log n)$, [x] $O(n)$, ☐ $O(n \log n)$, ☐ $O(n^2)$, ☐ $O(n^2 \log n)$, ☐ $O(n^3)$, ☐ $O(n^3 \log n)$, ☐ $O(n^4)$, ☐ $O(n^4 \log n)$.

(d)
```
function possum(ar,w) { var n = ar.length; var m = w*w;
   return(n*m*sumn(ar)*position(ar,w)); }
```

Runtime is ☐ $O(1)$, ☐ $O(\log n)$, ☐ $O(n)$, ☐ $O(n \log n)$, [x] $O(n^2)$, ☐ $O(n^2 \log n)$, ☐ $O(n^3)$, ☐ $O(n^3 \log n)$, ☐ $O(n^4)$, ☐ $O(n^4 \log n)$.

(e)
```
function last(ar) { var n = ar.length; return(ar[n-1]); }
```

Runtime is [x] $O(1)$, ☐ $O(\log n)$, ☐ $O(n)$, ☐ $O(n \log n)$, ☐ $O(n^2)$, ☐ $O(n^2 \log n)$, ☐ $O(n^3)$, ☐ $O(n^3 \log n)$, ☐ $O(n^4)$, ☐ $O(n^4 \log n)$.

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

Write a Java Script function which counts the number of prime numbers below n. So count(7) is 4 as there are the four prime numbers 2, 3, 5 and 7 below 7. Similarly count(8) is 4 and count(11) is 5.

```
function count(n)
  { var m;


    var ar = new Array(0,0);
    while (ar.length <= n) { ar.push(1); }
    var i; var j;
    for (i=2;i<=n;i++)
      { for (j=i+i;j<=n;j+=i)
          { ar[j]=0; } }
    m=0;
    while (ar.length>0) { m+= ar.pop(); }


    return(m); }
```

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

Assume that an array ar of length n is given. Write a Java Script function which finds the first m such that there are distinct i,j below m with ar[i] being equal to ar[j]. So if ar equals (2,5,4,5,2) then m is 3 as ar[1] and ar[3] are both 5. In the case that there are no two members of the array, the return value of the function is 0.

```
function firstm(ar)
  { var n = ar.length; var m = 0;


    var i; var j;
    for (j=n-1;j>0;j--)
      { for (i=0;i<j;i++)
          { if (ar[i]==ar[j])
              { m = j; } } }


    return(m); }
```

END OF PAPER