One and a half hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Algorithms and Imperative Programming

Date: Wednesday 20\textsuperscript{th} January 2010
Time: 14.00 – 15.30

Please answer the compulsory Question in Section A
and
ONE question from Section B

Please use separate Answerbooks for EACH section

The use of electronic calculators is NOT permitted
Section A

1. **Compulsory**

a) Consider the number of operations used by the following three algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Number of operations, when input size is ( n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original algorithm</td>
<td>( n^3 + 5n^2 )</td>
</tr>
<tr>
<td>Improved algorithm 1</td>
<td>( n^3 + 4n )</td>
</tr>
<tr>
<td>Improved algorithm 2</td>
<td>( n^3/2 + 4n )</td>
</tr>
</tbody>
</table>

What is the time complexity of the three algorithms, expressed in a big “O” notation? (3 marks)

If \( n \) is large, does any of the algorithms run significantly faster? Explain your answer. (2 marks)

b) The following functions all have different asymptotic complexities. Place them in increasing complexity order.

\[ 2^n, \sqrt{n}, 5n + 4n^2, n^3, n!, \log n, 7n \] (3 marks)

c) Neesha has developed an algorithm for a complicated problem. It is too difficult to determine the running time of her algorithm mathematically, so she decides to run some experiments.

She guesses that the algorithm has a running time of the form \( t(n) = bn^c \), and decides to perform a power test to confirm this.

Her results, plotted on a log-log plot are shown below.

(Question 1 continues on the following page)
Explain why, from looking at the plot, the running time can be expressed as \( t(n) = bn^c \). Which of the following words can be used to describe the running time: exponential, logarithmic, factorial, polynomial or linear? (4 marks)

Using the intercept and slope from the plot, estimate \( b \) and \( c \), and give an estimated formula for the running time in terms of \( n \). (3 marks)

d) Below is an algorithm for printing out all the prime factors of each positive integer from 1 to \( n \).

**Algorithm** PrimeFactors\( (n) \):

1. for \( i \leftarrow 1 \) to \( n \) do
2. \hspace{1em} for \( j \leftarrow 2 \) to \( i - 1 \) do
3. \hspace{2em} if testifprime\( (j) \) is true and \( j \) divides \( i \) then
4. \hspace{3em} printline\( (j \text{ “is a prime factor of” } i) \)

What is the time complexity of PrimeFactors\( (n) \) expressed using the big “O” notation if the function testifprime\( (x) \) is \( O(n) \)? (2 marks)

Here is a faster algorithm for the same task:

**Algorithm** FastPrimeFactors\( (n) \):

1. for \( i \leftarrow 1 \) to \( n \) do
2. \hspace{1em} for \( j \leftarrow 2 \) to \( \lfloor \sqrt{i} \rfloor \) do
3. \hspace{2em} if \( j \) divides \( i \) then
4. \hspace{3em} if testifprime\( (j) \) is true then
5. \hspace{4em} printline\( (j \text{ “is a prime factor of” } i) \)
6. \hspace{3em} if testifprime\( (i/j) \) is true then
7. \hspace{4em} printline\( (i/j \text{ “is a prime factor of” } i) \)

Reminder: \( \lfloor x \rfloor \) means the integer part of \( x \). It is equivalent to (int)\( x \) in the C programming language.

Explain why it is sufficient for the inner loop to stop at \( \lfloor \sqrt{i} \rfloor \). What is the time complexity of the algorithm? (3 marks)
2. **Hash Tables**

Consider the following sequence of numbers:

1, 22, 13, 88, 23, 39, 11, 6, 16, 5

We wish to insert these numbers in this order in a hash table of size 11 indexed from 0 through to 10 using the hash function \( h(x) = (2x + 5) \mod 11 \) (where “mod N” means the remainder after division by N). The resulting hash values of the sequence above is:

7, 5, 9, 5, 7, 6, 5, 6, 4, 4

a) Show the resulting hash table if collisions are handled by chaining, i.e. sequences at the same value. (2 marks)

b) Show the resulting hash table if collisions are handled by linear probing. Explain how the numbers are entered into the table. (3 marks)

c) Show the resulting hash table if collisions are handled by quadratic probing. Again, show how the numbers are entered into the table. Can all the numbers be inserted successfully? Explain your answer. (4 marks)

d) Time complexity: in the best case, inserting an element in a hash table, using linear probing, requires one operation (if the hash values are known): simply insert at the hash value. If collisions occur, more operations are required. How many operations are required to insert an element in a hash table of length N containing M elements in the worst case? Explain your answer. (4 marks)

e) Give a pseudo-code description (or a program outline in C or Java) for performing the removal of an element from a hash table that uses linear probing without using a special marker to represent deleted elements. Explain your answer. [Hint: one method is to insert some of the elements again using their hash value.] (7 marks)
3. **Trees**

   a) Insert the following items in a given order into an initially empty binary search tree:

   17, 23, 11, 32, 46, 9, 12, 37, 92.

   Draw the tree after each insertion.  

   (5 marks)

   b) Describe the algorithm for rebalancing an unbalanced binary search tree after inserting an element in the tree, using AVL trees. Repeat the insertion procedure from Part (a), and rebalance the tree at insertion steps where it is necessary, i.e. when the tree becomes unbalanced after an insertion.

   (10 marks)

   c) The lowest common ancestor between two nodes \( v \) and \( w \) in a binary tree is defined as the node which is the furthest away from the root and of which both \( v \) and \( w \) are descendants. Note that this definition allows for a node to be a descendant of itself. Given two nodes \( v \) and \( w \) in a binary tree, describe an efficient algorithm for finding the lowest common ancestor for them. Hint: one method is to use the Euler traversal of a tree with appropriate labelling of the nodes and derive the appropriate actions on these labels upon each visit to a node.

   (5 marks)

END OF EXAMINATION