UNIVERSITY OF LONDON

GOLDSMITHS COLLEGE

Department of Computing

B. Sc. Examination 2016

IS52038A
Algorithms and Data Structures

Duration: 2 hours 15 minutes

Date and time:

This paper is in two parts: part A and part B. You should answer ALL questions from part A and TWO questions from part B. Part A carries 40 marks, and each question from part B carries 30 marks. The marks for each part of a question are indicated at the end of the part in [.] *brackets.*

There are 100 marks available on this paper.

Electronic calculators must not be programmed prior to the examination. Calculators which display graphics, text or algebraic equations are not allowed.

THIS PAPER MUST NOT BE REMOVED
FROM THE EXAMINATION ROOM

Part A

Question 1

(a) Explain, with the aid of an example, the main differences between an *algorithm* and a (computer) *program*. [4]

(b) Company manager Alex thinks that algorithm efficiency is less important these days because the running speed of computers has increased dramatically. Do you agree with him? Concisely write about your view and give one convincing example to support your argument. [7]

(c) A sorting algorithm is regarded as *stable* if it maintains the relative order of identical data. Draw a sequence of diagrams to demonstrate that the insertion sort is stable, using (5, 2, 4, 6, 2, 7) as an input example. [5]

(d) Explain why, in the worst case, the number of comparisons is of $O(n^2)$ for the insertion sort. [6]

(e) Explain what distinguishes a *binary search tree* from a *binary min-heap*. Draw diagrams to demonstrate, step by step, how each of the data structures can be used to store the data (5, 7, 10, 6, 1, 14, 11, 2). Assume that both the binary search tree and the binary min-heap are empty initially, and the data is added in the order given. [10]

(f) Fill the missing words/sentences in each gap below: [8]

NP class is the class of ___(1)___ problems which can be solved in ___(2)___ time by a ___(3)___ computer.

NP-complete is the term used to describe ___(4)___ problems that are the ___(5)___ ones in NP class in the sense that, if there were a ___(6)___ algorithm for a NP-complete problem, then there would be a ___(7)___ algorithm for ___(8)___ problem in NP.

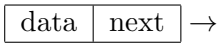
To prove a new problem is NP-complete, we first prove that ___(9)___, then we prove that ___(10)___.

Part B

Question 2

(a) Explain what is meant by *dynamic programming*. Demonstrate a case, with the aid of the example of the recursive *Fibonacci* algorithm, where *dynamic programming* is useful. [5]

(b) Consider design of the algorithm $mergelists(x, y)$ that merges two linked lists x and y. The data in both x and y are in ascending order. The result after a merge should be one linked list in ascending order, containing all the data in both lists. Assume the node structure is as follows and that you may modify only the *next* field of each node:



i. Assume that the result list is (0, 2, 3, 3, 4, 4, 5, 6, 7, 11, 14). Give an example of merging two linked lists including two instances of the problem: one must be a *general case* of the problem where x contains 5 elements, and the other one must be a *special case* of the problem. Explain your definition of the 'special case' if it is different from the one in lectures. [6]

ii. Devise and outline your algorithm $mergelists(x, y)$ in pseudocode or in a flow chart. Assume no predefined ADT is available so the *Node* class needs to be constructed if necessary. [14]

iii. What is the worst-case time complexity of your algorithm? Justify your answer. [5]

Question 3

(a) Draw diagrams to demonstrate the derived *trie* and *compressed trie* that store the following words: (stop, bid, bear, stock, bell, bull, sell, buy). Assume that the data is input in the order given. [6]

(b) Demonstrate, step by step, the comparisons performed by the Boyer-Moore pattern matching algorithm. Use the example in which the text `T` and the Pattern `P` are: [11]

```
T: B C B D B B C B E D B C B D B C B B C C
P: B C B D B C
```

(c) Outline, in diagrams, how the Heapsort algorithm works, using (11, 12, 14, 16, 13, 17) as an input example. [7]

(d) Suppose we want to store data (3,4,5,26,6,7,23,16,39,17,22,55) in the hash table H[0..22] and use the hash function $h(k) = k \bmod 23$. Demonstrate the content of the hash table using *linear probing*. Assume that the hash table is empty initially and the data is added into the hash table one by one in the order given. Are there any collisions? If yes, explain with an example how you would handle a collision. [6]

Question 4

(a) Consider the following algorithm `StackAndQueue`. What does it do essentially? Suppose that the stack `S` contains, initially, a '0' and the queue `Q` contains `4, 5, 6, 2, 3, 7` with 4 in front. What are the elements in `S` and `Q` on completion of the execution of the algorithm? [6]

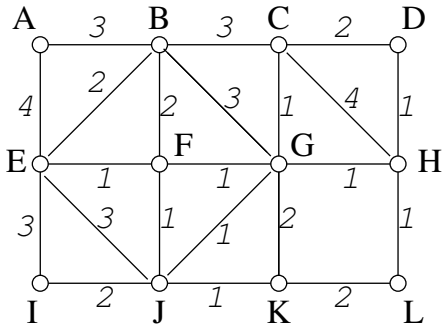
```
StackAndQueue(queue: Q);

stack S
integer obj
begin
  initialise(S);
  while not Empty(Q) do
    begin
      dequeue(Q,obj);
      push(S,obj);
    end
  while not Empty(S) do
    begin
      pop(S,obj);
      enqueue(Q,obj)
    end
end;
```

(b) Analyse and derive the time complexity of the `StackAndQueue(queue:Q)` algorithm in part (a). Show all your work. [5]

(c) What is the so-called 'greedy approach'? Describe the *matrix multiplication* problem and explain how this problem can be solved by a greedy approach. [5]

(d) Consider the instance of searching for a shortest path from vertex A to G of the simple weighted graph below. Assume that the estimated path distances from each vertex to G are, from A:2, from B:1, from C:1, from D:2, from E:2, from F:1, from G:0, from H:1, from I:2, from J:1, from K:1, and from L:2.



i. Write down the *weighted* adjacency matrix of the graph. [2]

ii. Demonstrate the *weighted* adjacency list of the graph. [2]

iii. Draw a series of diagrams to show how a search tree may be expanded step by step applying the Breadth-First Search algorithm for the problem instance. Highlight the characteristics of the algorithm and show all your work. [4]

iv. Demonstrate how the A* search algorithm works. Trace step by step the expansion of the search tree, the evaluation and choice made on each vertex visited in execution of the algorithm. Highlight the characteristics of the algorithm and show all your work. [6]