The University of Nottingham

SCHOOL OF COMPUTER SCIENCE

A LEVEL 2 MODULE, AUTUMN SEMESTER 2012-2013

ALGORITHMS AND DATA STRUCTURES

Time allowed TWO Hours

Candidates may complete the front cover of their answer book and sign their desk card but must NOT write anything else until the start of the examination period is announced

Answer QUESTION ONE and THREE other questions

Clearly cross through any question that you do NOT wish to be considered and ensure you state on the front of the answer book the FOUR questions that you have attempted. Marks available for sections of questions are shown in brackets in the right-hand margin

Only silent, self-contained calculators with a Single-Line Display are permitted in this examination.

Dictionaries are not allowed with one exception. Those whose first language is not English may use a standard translation dictionary to translate between that language and English provided that neither language is the subject of this examination. Subject specific translation dictionaries are not permitted.

No electronic devices capable of storing and retrieving text, including electronic dictionaries, may be used.

DO NOT turn your examination paper over until instructed to do so

ADDITIONAL MATERIAL: none

INFORMATION FOR INVIGILATORS: Students should write all their answers in the answer book. The exam paper should be collected and placed inside the answer book.

1. This multiple choice question is compulsory. For each part select precisely **one** answer. You will not receive negative marks for incorrect answers. Write your answers clearly, in the answer book, in the form of a single table with one answer per row and including the label for which part of this question you are answering, and the answer in upper-case, i.e. in the form "(x) Y" per row.

(a) The function $3\,n^2 + 4\,n\log(n)$ is

A) $\Theta(n^2)$ but not $O(n^3)$ (big-Oh)

B) $\Omega(n^2)$ and $o(n^2)$ (little-oh)

C) $\Theta(n^2 / \log(n))$

D) $\Theta(n^2)$ and $o(n^3)$

E) $\Theta(n \log(n))$

(4 marks)

(b) Starting from an empty binary search tree, entries with keys 5, 7, 3, 4, 8, 6 are inserted in that order. A subsequent **pre-order** traversal of the resulting tree will produce the keys in which of the following sequences:

A) 3, 4, 5, 6, 7, 8

B) 4, 3, 6, 8, 7, 5

C) 5, 3, 4, 7, 6, 8

D) 5, 3, 7, 4, 6, 8

E) 5, 3, 4, 7, 6, 8

(4 marks)

(c) Consider a binary tree satisfying the heap properties and containing n nodes, then the height of the tree is

A) $o(\log n)$ (little-oh)

B) $\Theta(\text{sqrt } n)$

C) $O(\log n)$ but not $\Omega(\log n)$

D) $\Omega(n)$

E) $\Theta(\log n)$

(4 marks)

(d) Consider the following code for insertion sort to sort an array of integers into non-decreasing order (note that the array indices start at 0 as usual)

```
1.  void sort( int[] arr ) {
2.              int i, k, temp;
3.              for ( k=1 ;  k < arr.length ; k++ ) {
4.                      assertTrue( S );
5.                      temp = arr[ k ];
6.                      i = k;
7.                      while( i > 0 && arr[i-1] >= temp ) {
8.                              arr[i] = arr[i-1];
9.                              i--;
10.                     }
11.                     arr[i] = temp;
12.             }
13.     }
```

What is the most appropriate and useful choice for assertion S on line 4:

A) i >= 1

B) i >= 1 && k >= 1

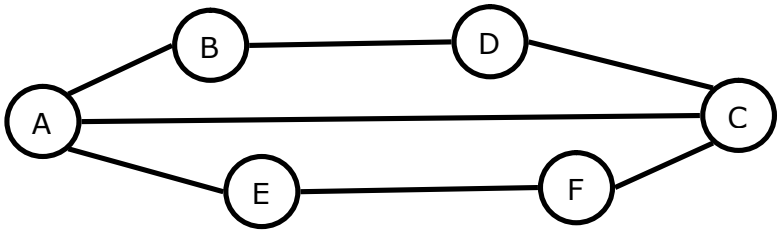
C) forall a such that 2 <= a < k. (arr[a-1] <= arr[a])

D) forall a such that 1 <= a < k. (arr[a-1] >= arr[a])

E) forall a such that 1 <= a < k. (arr[a-1] <= arr[a])

(4 marks)

(e) Consider the following graph



Which of the following corresponds to a possible Depth-First Traversal starting from node A?

A) A, C, D, B, F, E

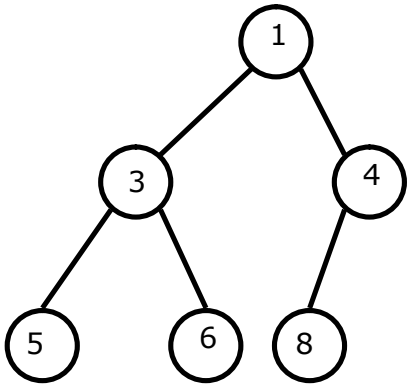
B) A, B, D, E, F, C

C) A, E, B, D, C, F

D) A, C, D, F, B, E

E) None of the above (4 marks)

(f) The following binary tree is the initial state of a heap



Do a removeMin() operation to remove the minimum key, 1, then insert an entry with key 2, and finally convert to the array based representation in the standard fashion used for heaps, that is, with the root of the heap placed at index 1, etc. Which of the following is the final state of the array? (Where a '#' denotes an unused or blank entry).

A) [# 2 4 3 # 8 6 5]

B) [# 2 3 4 5 6 8 #]

C) [# 2 5 3 8 6 4 #]

D) [# 2 3 4 # 5 6 8]

E) None of the above (5 marks)

2. This question is concerned with the 'big-Oh' family.

(a) Give, and also explain and justify, the definitions of big-Oh, big-Omega, big-Theta and little-oh by completing each of the following:

i) 'big-Oh': f(n) is O(g(n)) ….

ii) 'big-Omega': f(n) is Ω(g(n)) ….

iii) 'big-Theta': f(n) is Θ(g(n)) ….

iv) 'little-oh': f(n) is o(g(n)) ….

Your justification and explanation should address the most important parts of each definition showing why they are defined in the way that they are, and in particular, the choices of the quantifiers "there exist" and "for all" and their relative ordering.

(14 marks)

(b) From the definitions prove or disprove that $f(n) = 2\,n^3+n$ is $\Theta(n^3)$ (big-Theta).

(6 marks)

(c) From the definitions prove or disprove that $f(n) = 2n(\log n)$ is $o(n(\log n)^2)$ (little-oh).

(5 marks)

3. This question concerns the Vector data structure and amortised complexity.

(a) Explain and define the "Vector" Abstract Data Type, and give an example of when it would be useful.

(4 marks)

(b) Discuss the notion and usage of amortised complexity when considering a sequence of operations on a data structure. In particular, be clear about the difference from the worst case complexity.

(4 marks)

(c) Consider a Vector implemented using an underlying array. When the array is full then it needs to be resized. For each of the following two schemes for increasing the array size give the process that is followed when resizing, and also give the worst case complexity and compute the amortised complexity:

i) Increase the array size by adding a constant K to the size when resizing.

ii) Increase the array size by doubling it when resizing.

(12 marks)

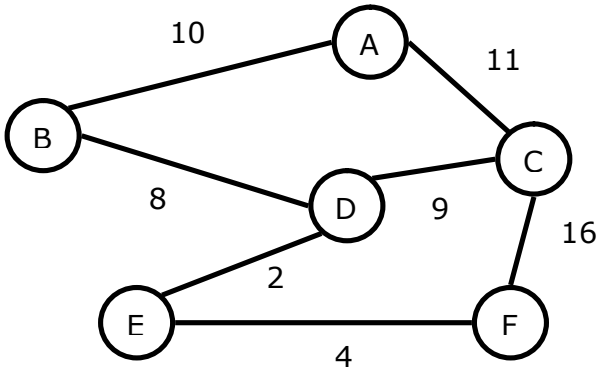
(d) Heaps are often discussed as being implemented using an array, resulting in a worst case insertion cost being O(log n). To allow an arbitrary size of heap, they could instead be implemented using a Vector. What are what the implications of part (c) for the worst-case and amortised complexity costs of insertion into a heap? Justify your answer.

(5 marks)

4. This question concerns paths and trees in graphs.

(a) Explain, and give pseudo-code for, Dijkstra's algorithm to find the shortest path in an undirected graph when the distances for edges are given and all are strictly positive (greater than zero). Include at least one appropriate assertion or loop invariant within your code. In particular, explain why the algorithm is guaranteed to give a shortest path.

Use Dijkstra's algorithm to find the shortest path from node A to node F in the graph below. Show your working.



(12 marks)

(b) Define and explain the concept of a "Minimum Spanning Tree" (MST) in an undirected weighted graph, and give the standard greedy Prim's algorithm for finding the MST. Also, compare the MST problem with the shortest path problem by giving a graph in which the MST is not a path.

(8 marks)

(c) Suppose that the nodes of a graph are considered as cities and that they have a width w which is the distance taken to travel from one side to the other. So, for example in the graph of part (a) travelling from A to F through C would cost 11 + w + 16.

Show how Dijkstra's algorithm can be modified to account for these node widths. Use your modified algorithm to re-compute the shortest path from node A to F in the graph of part (a) but with each node having width w = 3.

(5 marks)

5. This question concerns Maps and hash tables.

(a) Describe how hash tables can be used to implement the Map Abstract Data Type (ADT).
(5 marks)

(b) Consider linear probing for the hash table. Explain the issue of collisions, and how linear probing handles them. Explain how the 'insert' and 'remove' operations should work, and their typical and worst-case complexities.
Illustrate your answer by showing how they work in the following example:

With a hash table array of size N=7 and hash function

$$h(x) = (3\ x + 5\ ) \bmod N$$

The following sequence of 5 operations are performed starting from an empty table (and using linear probing):

insert 6
insert 4
insert 8
insert 1
remove 6
remove 1

Hint: Firstly, provide the hash value for each of the numbers inserted. Then use this to provide the state of the hash table after each operation, being sure to explain your working.
In your answer, write the array in the form [...] and denote an empty entry using the symbol "#" , that is, the array starts out as [# # # # # # #]
(15 marks)

(c) When the hash table becomes too full then it is necessary to resize the array. Explain the issues involved in such resizing; two methods that might be used to achieve this and their relative costs and benefits. In particular, be clear about how the resizing affects the typical and worst case complexities, and why these matter.
(5 marks)

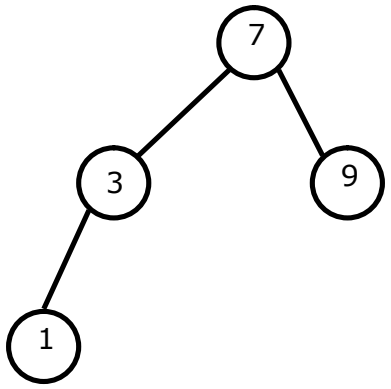
6. This question concerns Maps and Binary Search Trees.

(a)

i) State and briefly explain the standard methods of the interface of Map Abstract Data Type (ADT).
ii) State what it means for a binary tree to have the binary search tree property and why this makes it suitable to implement a Map.
iii) Explain the ways in which the heap property differs from the binary search tree property.

(5 marks)

(b) The following diagram shows the initial state of a binary search tree (BST).



Show and explain your working for the resulting state of the BST after performing each step of the following sequence of operations:

i) insert(5)
ii) insert(4)
iii) insert(8)
iv) delete(9)
v) delete(3)

(13 marks)

(c) It is desired to create an "adaptive priority queue" in which it is permitted to update a key. Specifically assume that all keys are unique and it is desired to extend a heap so as to be able to support a method "updateKey(int k1, int k2)" in which an entry with key k1 (if it exists) is changed to key k2 and the structure of the heap is then fixed if needed.

i) Discuss the difficulties of doing this directly and retaining the standard O(log n) complexity of heap operations.

ii) Discuss whether or not a combination of the heap with a separate data structure such as a BST or hash table might allow the updateKey operation to be implemented and retain the desired O(log n) complexity of all operations.

(7 marks)