JFK
38

THE UNIVERSITY OF THE WEST INDIES
ST. AUGUSTINE

EXAMINATIONS OF JULY 2018

Code and Name of Course: **COMP 2611 – Data Structures**

Date and Time: Friday 20th July 2018 1pm Duration: **2 hours**

INSTRUCTIONS TO CANDIDATES: This paper has **4** pages and **3** questions

The use of non-programmable scientific calculators is allowed.
Answer ALL Questions

PLEASE TURN TO THE NEXT PAGE

1. **(a)** A function **makeHeap** is passed an integer array **A**. If **A[0]** contains **n**, where n is the number of elements in the heap, then **A[1]** to **A[n]** contain numbers in arbitrary order.

   (i) Write **makeHeap** such that **A[1]** to **A[n]** contain a *min*-heap (*smallest* value at the root). Your function must create the heap by processing the elements in the order **A[2]**, **A[3]**,..., **A[n]**. [6 marks]

   (ii) If the array **A** contains the following values initially (remember **A[0]** is the number of elements in the heap):

8	30	15	12	25	21	10	33	13
0	1	2	3	4	5	6	7	8

   Show the contents of **A[1]** to **A[j]** after element **A[j]** (**j = 2..8**) is processed. [3 marks]

   (iii) Given an array like **A**, with **A[1]** to **A[n]** containing a *min*-heap, write a function to sort the array in *descending* order. [6 marks]

   (b) You are given the following postorder and inorder traversals of a binary tree.

   Postorder F E C H G D B A
   Inorder F C E A B H D G

   Draw the binary tree. [3 marks]

   [Total marks: 18]

2. **(a)** Each node of a *binary search tree* has fields **left**, **right**, **key** (an integer) and **parent**, with the usual meanings. Write a function which, given a pointer to the root of the tree and an integer **n**, searches for **n**. If found, return a pointer to the node. If not found, add **n** to the tree, ensuring that *all fields* are set correctly. Return a pointer to the new node. You may assume that the function call **newNode(n)** creates a node, stores **n** in it and returns a pointer to the node. [8 marks]

PLEASE TURN TO THE NEXT PAGE

(b) In a hashing application, the key consists of a string of letters. Write a hash function which, given a key and an integer `max`, returns a hash location between 1 and `max`, inclusive. Your function must use *all* of the key and should not deliberately return the same value for keys consisting of the same letters. [3 marks]

(c) A hash table of size `n` contains two fields—an integer data field and an integer link field (called *data* and *next*, say). The *next* field is used to link data items in ascending order. A value of -1 indicates the end of the list. The variable `top` (initially set to -1) indicates the location of the smallest data item.

Integers are inserted in the hash table using "open addressing with double hashing". Assume that the function `h1` produces the initial hash location and `h2` produces the increment. An available location has the value `Empty` and no item is ever deleted from the table.

Write programming code to search for a given value `key`. If found, do nothing. If not found, insert `key` in the table and *link it in its ordered position.* You may assume that the table contains room for a new integer. [8 marks]

[Total marks: 19]

3. (a) An `n` × `n` matrix `A` is used to store the points obtained in football matches among `n` teams. A team gets `3` points for a win, `1` point for a tie and `0` points for a loss. `A[i,j]` is set to `3` if team `i` beats team `j`; it is set to `1` if the match is tied and it is set to `0` if team `i` loses to team `j`.

In order to conserve storage, the values in the (strictly) lower triangle of `A` are stored in an array `B[1..m]` in row order.

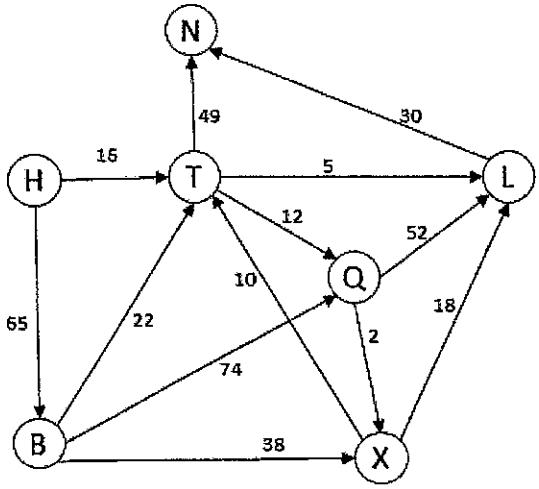
(i) What is the value of `m` in terms of `n`? [1 mark]

(ii) Write a function `score(i,j)` which, by accessing `B`, returns the value of `A[i,j]`. If `i` or `j` is invalid, the function returns `-1`. [5 marks]

(iii) Using the function in (ii), write another function which, given `t`, returns the total number of points earned by team `t`. [3 marks]

PLEASE TURN TO THE NEXT PAGE

(b) Given the following graph:



(i) Draw the adjacency list representation of the graph. List nodes in alphabetical order. [2 marks]

(ii) Give the depth-first and breadth-first traversals of the graph starting at `B`. Edges leaving a node are processed in alphabetical order. [2 marks]

(iii) Make a copy of the graph *without* the edge weights. Assume that a depth-first traversal is performed starting at `B` and that *edges of a node are processed in alphabetical order*. Indicate the discovery and finish times for each node and label each edge with `T` (tree edge), `B` (back edge), `F` (forward edge) or `C` (cross edge), according to its type. [5 marks]

(iv) State, with a reason, whether or not it is possible to topologically sort the nodes of the graph. [1 mark]

(v) Assuming that the graph in (a) is undirected, draw the minimal spanning tree obtained by using Kruskal's algorithm. Show the steps in your derivation. [4 marks]

[Total marks: 23]

End of Question Paper