JFK
(66)

THE UNIVERSITY OF THE WEST INDIES
ST. AUGUSTINE

EXAMINATIONS OF DECEMBER 2017

Code and Name of Course: **COMP2611 – Data Structures**

Date and Time: Monday 18th December 2017 1 pm Duration: **2 Hours**

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

Answer ALL Questions

1. (a) The following are the *pre-order* and *in-order* traversals of the nodes of a binary tree:

Pre-order: H C P G T A N R E X L
In-order: P C T G A H R N X E L

Draw the tree. [4]

(b) 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. [5]

(c) Write a function `deleteLargest` which, given a pointer to the root of a *binary search tree*, deletes the node containing the *largest* value and returns a pointer to the root of the (new) tree. Your function must work for all cases. [3]

(d) The integer elements of an almost complete binary tree are stored in an array `A[1..n]`, with the root in location `1`.

(i) *Without sorting*, write a function to rearrange the elements of `A` so that the *largest* integer is in location `1`, and the largest integer of each subtree is also in the root of that subtree. The elements are to be processed in the order `A[2]`, `A[3]`, and so on, up to `A[n]`. [5]

(ii) If the array `A` contains the following values initially (`n` = 8):

```
53  36  25  47  61  32  75  59
```

Show the contents of `A[1]` to `A[i]` (`i` = 2 to 8) after each element `A[i]` is processed. [3]

Total marks Q1: 20

2. (a) 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]

(b) In a certain application, keys which hash to the same location are held on a linked list. The hash table location contains a pointer to the first item on the list and a new key is placed at the end of the list. Each item in the linked list consists of an integer `key`, an integer `count` and a pointer to the next element in the list. Storage for a linked list item is allocated as needed. Assume that the hash table is of size n and the call `H(key)` returns a location from `1` to n, inclusive.

(i) Write programming code, including all relevant declarations, to initialize the hash table. [2]

(ii) Write a function which, given the key `k`, searches for it. If not found, add `k` in its appropriate position and set `count` to `0`. If found, add `1` to `count`; if count reaches `10`, delete the node from its current position, place it at the head of its list and set `count` to `0`. [7]

(c) 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.

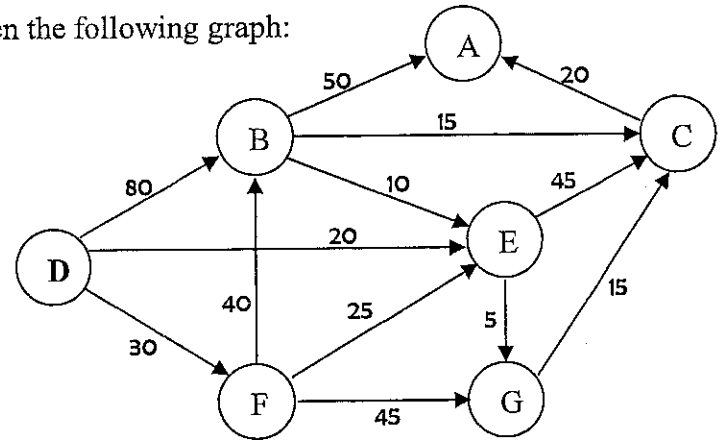
(i) What is the value of m in terms of n? [1]

(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`. [4]

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

Total marks Q2: 20

3. (a) Given the following graph:



(i) Give the depth-first and breadth-first traversals of the graph starting at **D**. Edges of a node are processed in alphabetical order. [2]

(ii) Derive the minimal-cost paths from node D to every other node using Dijkstra's algorithm. For each node, give the cost and the path to get to the node. At each stage of the derivation, show the minimal cost fields, the parent fields and the priority queue. In your table heading, list the nodes in *alphabetical order*. [8]

(b) A directed graph `G(V, E)` is represented using *adjacency lists*. The graph has *n* nodes stored in an array `G[1..n]`. Each node, `G[i]`, has the following fields:

```
char id;      //the name of the node; a single character e.g. A, F, T
int colour;
int parent;   //an array subscript indicating the location of a parent node
GEdgePtr first; //pointer to the first edge from the node; null, if none
```

Each edge node has the following fields:

```
int child;     //a subscript in G; G[child] is the child node
int weight;
GEdgePtr next; //pointer to the next edge; null, if none
```

(i) Write a function which, given `G` and `n`, outputs the graph, listing the nodes in the order in which they appear in `G`. Each node is followed by the name and weight of the edges leaving it. [3]

(ii) Write a function which, given `G`, `n` and a node name `S`, performs a depth-first traversal of `G` starting at `S`. Output the name of a node as it is visited and set the `parent` fields so that a depth-first path can be determined. Assume that all nodes are reachable from `S`. [7]

Total marks Q3: 20

END OF EXAMINATION