JFK
59

THE UNIVERSITY OF THE WEST INDIES
ST. AUGUSTINE

EXAMINATIONS OF December 2011

Code and Name of Course: COMP2000 – Data Structures

Date and Time: Monday 19th December 2011 1pm. Duration: 2 Hours

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

Answer ALL Questions

1. (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 another hashing application, keys which hash to the same location are held on a linked list *sorted in ascending order* with the hashtable location containing a pointer to the first item on the list. Each item in the linked list consists of an integer key and a pointer to the next element in the list. Use the names *key* and *next* for these fields. Storage for a linked list item is allocated as needed. Assume that the hash table is of size 53.
   Write programming code, *including all relevant declarations*, to

   (i) initialize the hash table; [2]

   (ii) search for the number *m*. If found, return the pointer to the node containing *m*. If not found, insert *m* in its appropriate position and return a pointer to the new node. [6]

   (c) Given a pointer to the root of a binary search tree of integers, write programming code to delete the root and return a pointer to the root of the new tree. If the root has non-empty left and right subtrees, replace it by its *inorder predecessor*. Assume that each node of the tree contains 3 fields – **data**, **left** and **right**. [6]

   (d) In a binary tree of *n* nodes, an 'external' node is attached to each null pointer. If I is the sum of the levels of the (internal) nodes of the tree and E is the sum of the levels of the external nodes, prove that $E - I = 2n$. [3]

   (e) Write a structured, non-recursive algorithm to traverse a binary tree in post-order. The only pointer fields in each node are **left** and **right**. You may assume the existence of the usual functions for manipulating a stack. [5]

The rest of this page is intentionally blank

2. (a) An *integer* max-heap is stored in an array (**A**, say) such that the size of the heap (**n**, say) is stored in **A[0]** and **A[1]** to **A[n]** contain the elements of the heap with the *largest* value in **A[1]**.

   (i) Write a function **deleteMax** which, given an array like **A**, deletes the largest element and reorganizes the array so that it remains a heap. [5]

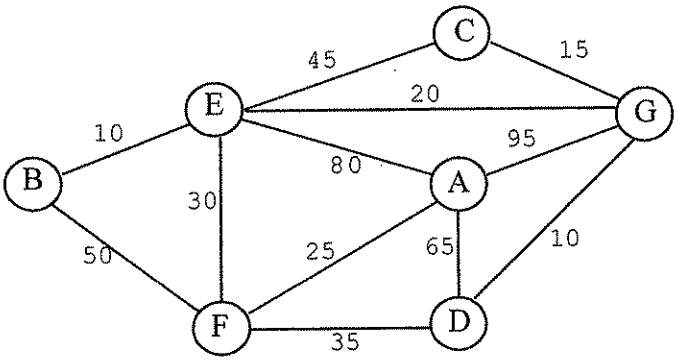
   (ii) Given two arrays **A** and **B** containing heaps as described above, write programming code to merge the elements of **A** and **B** into another array **C** such that **C** is in ascending order. Your method must proceed by comparing an element of **A** with one in **B**. You may assume that **deleteMax**, above, is available. [8]

   (b) A function **makeHeap** is passed an integer array **A**. If **A[0]** contains **n**, then **A[1]** to **A[n]** contain numbers in arbitrary order. Write **makeHeap** such that **A[1]** to **A[n]** contain a max-heap (*largest* value at the root). Your function must create the heap by processing the elements in the order **A[2]**, **A[3]**,...,**A[n]**. [6]

   (c) 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. [6]

The rest of this page is intentionally blank

(3) (a) Given the following undirected graph:



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

(ii) Derive the minimal-cost paths from vertex **A** to every other vertex using Dijkstra's algorithm. For each vertex , give the cost and the path to get to the vertex . At each stage of the derivation, show the minimal cost fields, the parent fields and the priority queue. [8]

(iii) Draw the minimum spanning tree obtained using Prim's algorithm, starting with vertex **A**. At each step, identify the edge selected for adding to the tree. [3]

(b) Let **A** be the adjacency matrix representation of the graph in 3 (a), assuming that each letter node is represented by its position in the alphabet. **A[j, j] = 0** for all **j**. If there is no edge from vertex **i** to vertex **j**, let **A[i, j] = 999**.

(i) Explain how **A** can be stored in a one-dimensional array **B[1..m]** conserving storage as much as possible. Assuming that **A** is an **n x n** matrix, what is the value of **m** in terms of **n**? [3]

(ii) Write a function which, given a node **j**, accesses **B** and returns the sum of the weights of edges leaving **j**. It may be helpful to write another function which returns the weight of an edge from node **i** to node **j**. [4]

(c) Write an algorithm to to determine if a directed graph is acyclic. Hint: use a modified depth-first traversal. [5]

End of question paper