CS 112 Data Structures

Midterm Exam 1

Fall, 2000

Circle your instructor/TA combination:

Ikro Yoon / Rajesh Bhowmick	Srikrishna Divakaran / Kiran Nagaraja
Lou Steinberg,/ Haiyan Cao	Barry Wittman /, Haiyan Cao
Lou Steinberg,/ Steven Sanbeg	Barry Wittman / Prashant Shah

- Count the pages in this exam. There should be 5 pages including this front page.
- Fill in your name and ID number above.
- Put your name on **each page**.
- Some questions are intended to be hard enough to challenge the our most able students, while others are intended to be easy enough for anyone in the class to do. If a problem seems very hard or is taking a long time, *don't panic* - just go on to other problems, and remember that the course is graded on a curve.

FOR GRADER'S USE ONLY:

1 _____ /25

2 _____ /25

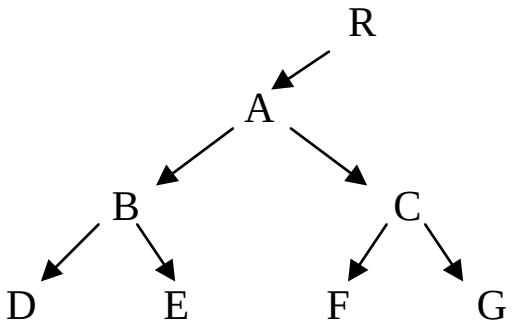
3 _____ /25

4 _____ /25

TO TAL _____ /100

1. General Trees and Forests; non-recursive traversal

Assume the following binary tree represents a general tree in the standard way.



A. Draw the general tree it represents

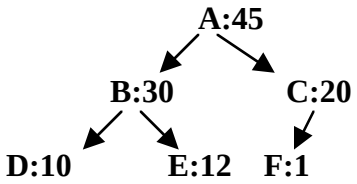
B. The following pseudo-code is supposed to do a breadth-first traversal of a general tree, assuming the general tree has been represented as a binary tree. Fill in the blanks.

```
traverse(root){
Queue q = new Queue;
enter root on q;
while (not (empty(q)){
      Node node = dequeue(q);
      print node.data;
      ptr = node.left;

      while(_____________){

            enter  _________________ on s;

            ptr = _____________________;
      }
}
```

2. Heaps, priority queues, and heapsort

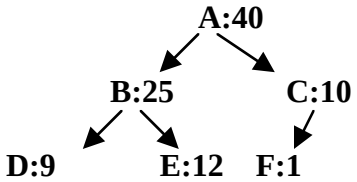
Suppose we have a program that plays music on request. At any time, it can receive a request to play a song. It assigns a priority to the request, and saves it in a priority queue. Whenever a song finishes playing, the program finds the request in the queue that has the highest priority and plays that request. Assume no two requests have equal priority. Below are pictures of a heap that is being used to implement the priority queueA node like A:3 means song A, priority 3.

A. Draw the following heap as it would look if the following requests were added **in this order:** G:25, H:9

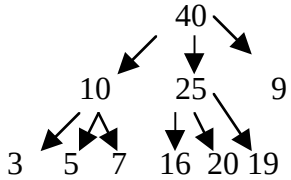


B. Suppose we removed the next request from the following heap. List the first request that would be removed, and draw the heap as it would look after this request is removed.

Request _____



C. Suppose we made heaps where each node had 3 children, rather than the normal 2 children, e.g.

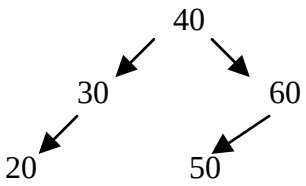


Does this increase the amount of work it takes to insert a new node in the heap, decrease it, or leave it the same? Why? We are concerned here with the actual amount of work, **not** the big-O. Assume worst case.

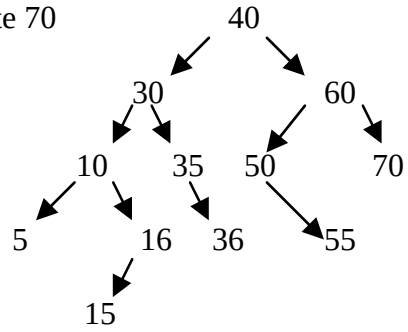
3. AVL trees

For each of the following AVL trees, redraw it to show how it would look after the insertion or deletion indicated.

A. insert 10



B. delete 70



4. Graphs

Here are three representations of an UNDIRECTED simple graph with no edge weights. In all the representations, the vertices are numbered 1 through N where N is the number of vertices in the graph. M is a number larger than N; it determines the amount of storage devoted to the representation(s), as described below.

Adjacency matrix (AMAT): graph is stored in the form of an MxM boolean matrix, A. The entries A[i,j] and A[j, i] are **true** The if there is an edge (i, j) in the graph; otherwise they are **false.**

Adjacency lists (ALL): The graph is stored as an array A (of size M) of linked lists. The entry A[i] gives a pointer to a linked list which contains the vertex numbers of all the vertices adjacent to vertex i, , stored in NO PARTICULAR ORDER. An edge (i, j) in the graph will appear in two linked lists: vertex i will appear in A[j]'s linked list and vertex j will appear in A[i]'s linked list.

Adjacency AVL trees (AAVL): The graph is stored as an array A (of size M) of AVL trees. The entry A[i] gives a pointer to an AVL tree whose keys are the numbers of all the vertices adjacent to vertex i.

For <u>each</u> task below, give the WORST CASE time complexity in "big O" notation for <u>each</u> of the three graph representations (AMAT, ALL, AAVL). Assume that there are N nodes and E edges in the graph. Explain your answers briefly.

Delete edge (i, j) from the graph.

Count the number of nodes adjacent to node i.

Given vertices i and j, determine (return true or false) if there is a arc between them.