Family Name:..... **Other Names:** .....

Student ID:..... **Signature**.....

COMP 103: Quiz

2020, July 14

Instructions

- Time allowed: **24 Hours**
- Starts: Saturday July 14, 080:00 AM in China (12:00 PM NZ time)
- Finishes: Saturday July 15, 08:00 AM in China (12:00 PM NZ time). **Late submissions will receive Zero**
- Attempt **all** the questions.
- Write your answers in this exam paper and hand in all sheets.
- If you think some question is unclear, ask for clarification.
- Brief Java documentation is provided with the test
- You must submit your work in **ONE** PDF (*quiz.pdf*) file into the online submission system.

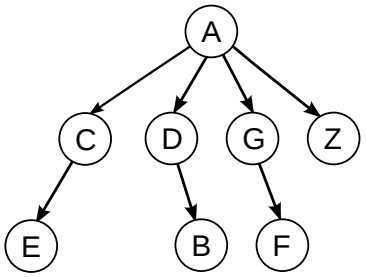
Question 1. General Tree Traversals **[13 marks]**

Consider the following traversalDFS method that does a depth first traversal of a general tree, printing out values in the nodes.

```
public void traversalDFS(GenTreeNode root){
    if (root==null) return;
    Stack<GenTreeNode> stack = new Stack<GenTreeNode>();
    stack.push(root);
    while (!stack.isEmpty()){
        GenTreeNode node = stack.pop();
        UI.print(node.getValue()+" ");
        for(GenTreeNode child : node.getChildren()){
            stack.push(child);
        }
    }
}
```

(a) **[2 marks]** Does traversalDFS do a pre-order or a post-order traversal of a tree? Justify your answer.

(b) **[4 marks]** List the output of the traversalDFS method if it were called on the following tree. Assume that the children of a node are shown in order from left to right.



(Question 1 continued on next page)

(Question 1 continued)

(c) **[7 marks]** Modify the traversal method above so that it does a breadth first traversal of a general tree instead of a depth first traversal. It should print the above tree in the order A C D G Z E B F.

```
public void traversalBFS(GenTreeNode root){
    if (root==null) return;









}
```

Question 2. Using collections **[28 marks]**

This question is about a program called SongsOrganiser, which manages information about a playlist of Songs. The following code for a Song class just stores information:

```
public class Song {
    private int year;
    private String artist, title;

    // constructor
    public Song(String artist, String title, int yr) {
        this.year = yr;
        this.artist = artist;
        this.title = title;
    }
    // some 'get' methods to provide information.
    public String getArtist() { return artist; }
    public String getTitle() { return title; }
    public Integer getYear() { return year; }
}
```

A real SongsOrganiser program would read in data from a large file. The demo version shown here creates Song objects explicitly, and stores them in a List of Songs:

```
public class SongsOrganiser {
    // constructor
    public SongsOrganiser() {
        List <Song> playlist = new ArrayList <Song> ();

        playlist.add(new Song("Phoenix Foundation", "Hitchcock",2005));
        playlist.add(new Song("Phoenix Foundation", "Hitchcock",2005));
        playlist.add(new Song("Phoenix Foundation", "Buffalo",2010));
        playlist.add(new Song("Dresden Dolls", "Shores of California",2007));
        playlist.add(new Song("Tiny Ruins", "Priest with balloons",2011));
        playlist.add(new Song("SJD","Lena",2012));
        playlist.add(new Song("Azelia Banks", "212",2011));
        playlist.add(new Song("SJD", "Beautiful haze",2007));
        playlist.add(new Song("SJD", "Beautiful haze",2007));
        playlist.add(new Song("Phoenix Foundation", "Buffalo",2010));

        // re-order the list
        reorderPlaylist(playlist);

        // remove duplicates
        playlist = removeDuplicates(playlist);
    }
    :
}
```

(Question 2 continued on next page)

(Question 2 continued)

(a) **[10 marks]** In the box below complete the code for reorderPlayList so that it sorts the Songs using a comparator based on their year, artist, and title, *in that order*. That is, if two Songs differ in their year it should order them on that basis, but if the years are the same, it should compare them by their artist field using the usual alphabetical ordering. And if they still agree, it should compare the titles.

Note: in Java, strings are Comparable: a String can call the "compareTo(String another)" method. This compares two strings lexicographically (ie. the usual alphabetical ordering).

```
public void reorderPlaylist(List<Song> playem) {




}
```

(Question 2 continued on next page)

(Question 2 continued)

(b) **[12 marks]** The playlist may have duplicates in it, which we would like to remove while preserving the order.

Write a method removeDuplicates for the SongsOrganiser class which takes a List of Song objects, and returns a List that is in the *same order*, but has any duplicates removed. removeDuplicates should not assume that the list has been sorted.

```
public void removeDuplicates(List<Song> songs){

}
```

Question 3. Binary Trees and Traversals **[28 marks]**

For this question, we define the depth of a tree to be the number of levels in it (so an empty tree has depth 0, a tree with a root and only one leaf to has depth 2, etc.)

A "full" binary tree is one in which all the levels of the tree are full — every node in the tree has two children, except for the nodes in the bottom level of the tree, which have none. An "unbalanced" binary tree is one in which some levels are not full, and some paths from the root to a leaf are much shorter than the paths to other leaves.

(a) **[2 marks]** What is the depth of a full binary tree that contains three nodes?

(b) **[3 marks]** Now consider a full binary tree whose depth is *one greater* than in the previous question. How many nodes are in this tree?

(c) **[4 marks]** How many leaves are there in the bottom layer of a tree with depth 7?

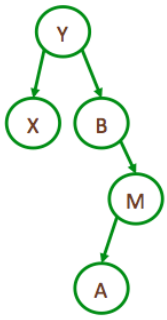
(d) **[2 marks]** What is the depth of a full binary tree with n nodes?

(e) **[2 marks]** What is the maximum depth of an *unbalanced* binary tree with n nodes?

(f) **[2 marks]** Roughly how many nodes are in a full binary tree with 32 layers (depth = 32).

(g) **[4 marks]** What will the following method print if called on the root of the tree shown below?

```
void treeprint(BinaryTreeNode node) {
    System.out.println(node.value);
    if (node.leftChild != null) treeprint(node.leftChild);
    if (node.rightChild != null) treeprint(node.rightChild);
}
```



(h) **[10 marks]** A "difference tree" is a binary tree in which each node contains an integer, every node in the tree has either two children or no children (a leaf), and every non-leaf node contains a value that is the difference between the value in its left child and the value in its right child.

Here is a partial description of a node in that tree:

```
public class TreeNode {
  public int getValue()
  public void setValue(int value)
  public TreeNode getLeftChild()
  public TreeNode getRightChild()
}
```

Complete the fixDifcTree(..) method so that it will turn a tree into a valid "difference tree", by changing the values in the non-leaf nodes. Assume that every node has either two children or no children, but the values in the non-leaf nodes may be wrong.

Note, fixDifcTree(..) returns the value in the node in its parameter..

```
  public int fixDifcTree(TreeNode root){

  }
```

Question 4. Traversing Graphs **[11 marks]**

Suppose you have a graph representing all the ISP's in a country, along with their high-bandwith fibre connections. Each ISP is represented by an Isp object that includes the name of the ISP, the number of clients, and a set of other Isps to which it has a direct high-bandwith fibre connection.

Complete the following printConnected method which will print out all the ISP's that are connected directly and indirectly to a given ISP. It should first print out the ISPs that are directly connected, then the ISPs that require two "hops", then the ISPs that reuqire three "hops", etc.

Assume that an Isp object is iterable so that you can iterate through the neighbours of an Isp node with

```
for (Isp neighbour : node){ ..... }
```

Hint: what kind of traversal will you need for this problem?

```
public void printConnected(Isp node){




}
```

* * * * * * * * * * * * * * *