[25 marks]

1 **(a)** Write Java code to implement each of the following methods for a *stack data structure* which uses a static array. List the class instance variables necessary for these methods. **[6marks]**

```
public ArrayStack (int size)  //  Constructor
public void Push(int S) //Add to the Top of
                        // the stack
public void Pop() //Remove the Top element of
                  // the stack
```

(b) Show using diagrams the result of adding the following 3 integer elements to an initially empty stack: **[4marks]**

```
Push(5);
Push(3);
Push(1);
```

Given the following implementation code for the *stack* methods `Top()` and `IsEmpty()`, explain how they operate on a *stack* data structure.

```
public int Top()
{
    if(IsEmpty()) {
        System.out.println("Stack empty \n");
        return TopOfStack;
        }
    else
        return SArray[TopOfStack];
}

private boolean IsEmpty(){return TopOfStack==-1;}
```

(b) Explain how a *stack* data structure may be implemented using the *ArrayList* data structure. **[5 marks]**

(c) What significant difference is there between a data structure created with the use of an array and one created with a linked list? Explain how a stack data structure may be implemented using a linked list. **[10 marks]**

[25 marks]

2 **(a)** Write Java code for each of the following methods for a Queue Class, *ArrayQueue*, to implement the operations of the *Queue* data structure using an array. List the class instance variables. **[6 marks]**

```
public ArrayQueue (int size)  //  Constructor
public void Enquque(int S) //Add to the back
                           // of the queue
public void Dequque() //remove from the front
                      //of the queue
```

(b) Show using diagrams the result of calling the following *ArrayQueue* methods from part (a) of this question. **[6 marks]**

```
ArrayQueue(3);
Enqueue(10);
Enquque(5);
```

Given the following implementation code for the *ArrayQueue* methods `getFront()` and `IsEmpty()`, explain how they operate on an *ArrayQueue* data structure.

```
public int GetFront()
{
  if(IsEmpty()) {
    System.out.println("GetFront Queue empty \n");
     return -1;
     }
 else
    return QArray[front];
}

private boolean IsEmpty(){return(Currentsize==0);}
```

(c) Explain, using diagrams, two methods of how a *queue* can be implemented as a circular array. Explain in pseudocode or Java code how the back and front references can be incremented in a circular array. **[7 marks]**

(d) Using the following Java *interface*, explain how other classes might implement this *interface*. **[6 marks]**

```
public interface Measurable
{
  double getMeasure();
}
```

[25 marks]

3 (a) Show each step in sorting the following list of numbers using *insertion sort*. **[7 marks]**

29 10 14 37 13

(b) Show using diagrams how nodes are added and removed from the front of a linked list data structure. Explain, using pseudocode, how the links to each node are updated. **[5 marks]**

(c) Write Java code to implement the following linked-list constructor and methods to add or remove the first element in the list. Write down the class instance variables necessary for the methods. **[7 marks]**

```
LinkedList()  //  Constructor
public Object removeFirst()
public Object addFirst()
```

(d) Explain the term 'Order-of-Magnitude' of an algorithm (Big-O notation). **[6 marks]**

[25 marks]

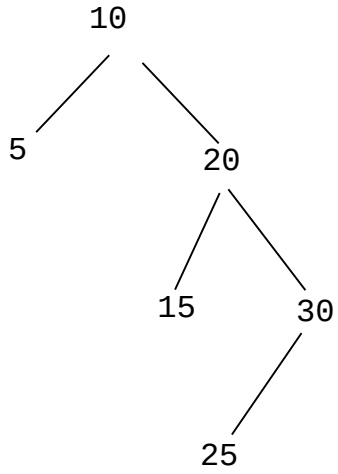
4 (a) Show the resultant binary tree after the following elements are inserted in the order given: **[12 marks]**

```
40   10  60  50  1  55  45
```

Show the order in which each of the following tree traversals visits the nodes in this tree:

(i) Pre-Order
(ii) In-Order
(iii) Post-Order

(b) Show the result of deleting node 20 from the following binary tree. **[7 marks]**



(c) Show how *mergesort* sorts the following list of numbers: **[6 marks]**

```
38 16 27 39 12 27
```

Explain what you understand by the divide-and-conquer approach used by mergesort.