1. (10 pt)

```
//function prototype for lengthOfArgument"
int argLength(int );



//Define a recursive function lengthOfArgumet here"

int argLength (int x)
{
      if (x/10 == 0)
            return 1;
      else
            return 1+ argLength(x/10);
}
```

2. (10 pt.)
   a. Tree – acyclic connected graph

   b. Forest – acyclic but disconnected graph

   c. Height of a tree – The largest depth of any node in tree T

   d. Binary Tree – a node has at most two children in a tree

   e. Binary search tree property – for each node, every left subtree node contain smaller value and every right subtree node contain larger value.

   f. what is stack – data structure with LIFO

   g. what is queue – data structure with FIFO

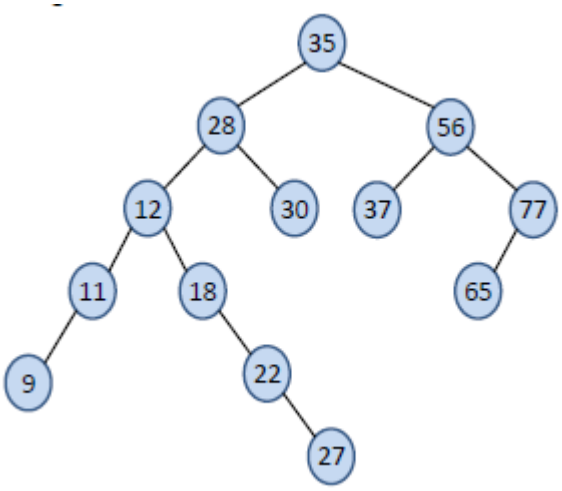
   h. What is composition of a class – an object can be a member of a class

   i. What is template? – the way to pass type as parameters

   j. What is static member in a class – only one copy which are shared by every instance of the class

3. (10 pt.)

35, 28, 56, 12, 30, 37, 18, 77, 22, 11, 27, 65, 9



4. (10 pt.) Following code shows recursive version of the Fibonacci series function. Write a non-recursive version of Fibonacci series function.

Non-Recursive version

```
// simple function without recursion
int fib (int x)
{
      int f2=1, f1=1, newTerm=0;
      if (x==0 || x==1)
            return 1;
      for (int i=2; i<=x; i++)
      {
            newTerm = f1 + f2;
            f2 = f1;
            f1 = newTerm;
      }
      return newTerm;
}
```

5. (5 pt)

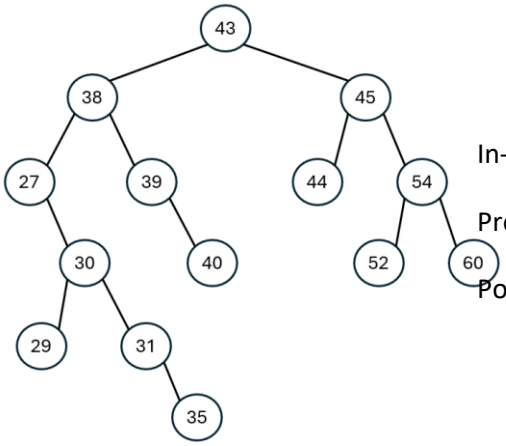
Hi ABC
Hi WXY
Bye WXY
Bye ABC

6. (10 pt.)

```
template<class DataType>
void Stack<DataType>::push(constDataType temp)
{
      // If stack is full, cannot push new node
      if (IsFull())
            cout<< "The stack is full.  Cannot Push().\n";
      else
{
            //create a node to push new data
            StackNode<DataType> *tempNode = new StackNode<DataType>;
            tempNode->data = temp;
            tempNode->next = top;
            top = tempNode;
            numNodes++;
      }
}
```

7.

a. **(10 pt.)What are outputs for the result of In-order, Pre-order and Post-order walk**.

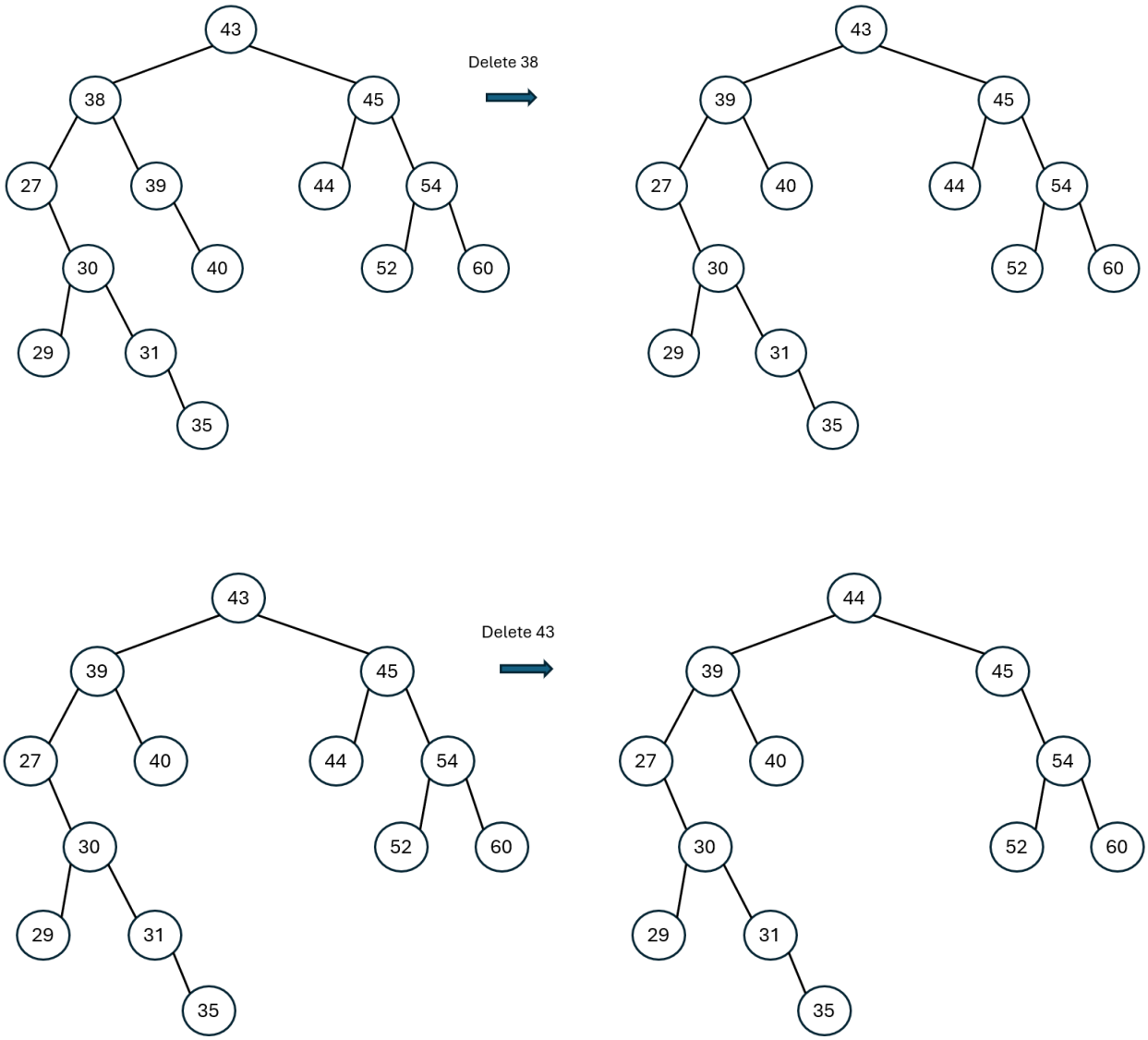


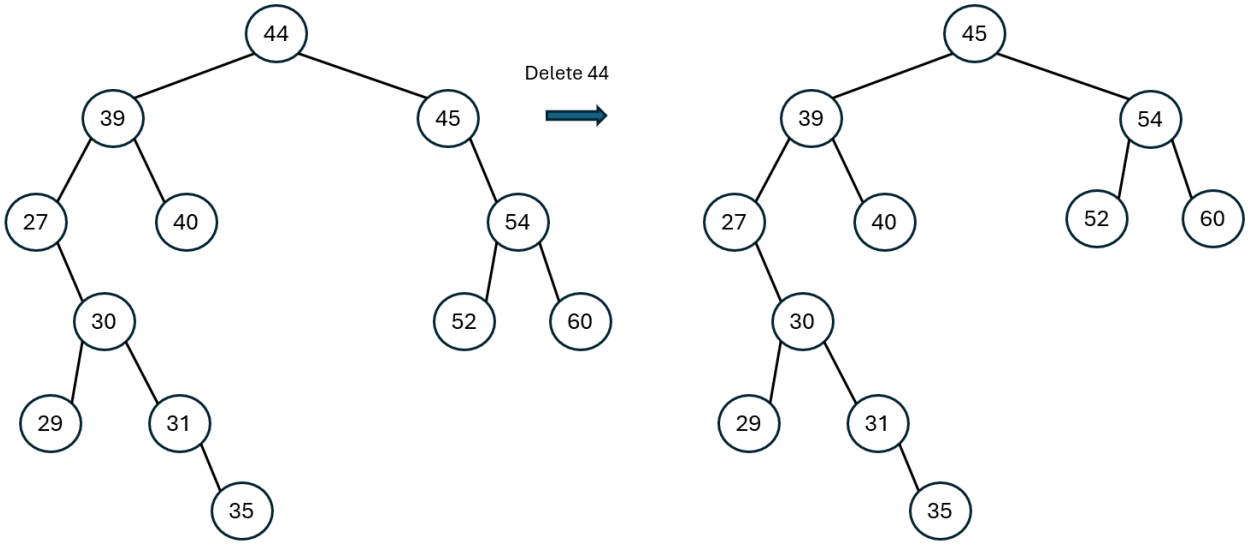
In-order- 27, 29, 30,31,35, 38, 39, 40, 43, 44, 45, 52, 54, 60

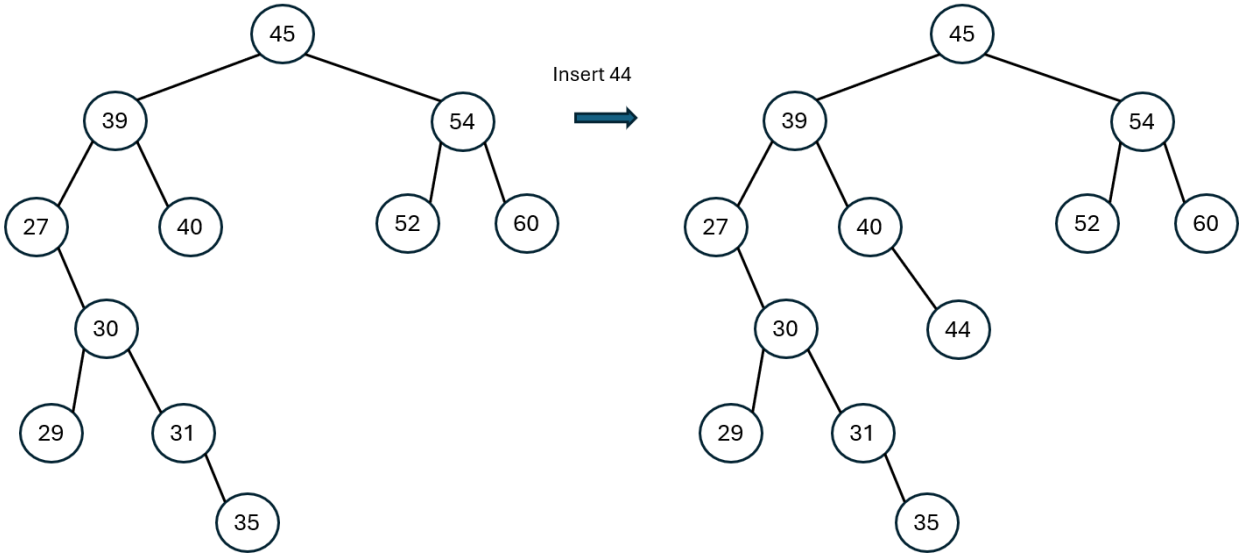
Pre-order –43, 38, 27, 30, 29, 31, 35, 39, 40, 45,44, 54, 52,60

Post-order –29, 35, 31, 30, 27, 40, 39, 38, 44, 52, 60, 54, 45,43

b. (10 pt.)


43
38
45
27
39
44
54
30
40
52
60
29
31
35
Delete 38
43
39
45
27
40
44
54
30
52
60
29
31
35
43
39
45
27
40
44
54
30
52
60
29
31
35
Delete 43
44
39
45
27
40
54
30
52
60
29
31
35


44
39
45
27
40
54
30
52
60
29
31
35
Delete 44
45
39
54
27
40
52
60
30
29
31
35


45
39
54
27
40
52
60
30
29
31
35
Insert 44
45
39
54
27
40
52
60
30
44
29
31
35

8. (15 pt.)

```
//a) function prototypes for overloading operators
       bool operator == (NameClass);

//define overloaded operator == here
bool NameClass::operator == (NameClass My)
{
       int count = 0;

       while ((Name[count] == My.Name[count])&& (Name[count] != '\0') && (My.Name[count]!= '\0'))
              count++;
       if (Name[count] != My.Name[count])
              return false;
       else
              return true;
}
```

9. (5 pt.) What are outputs for the following program?

```
This is A
This is A
This is B
This is A
This is B
This is C
3
3
3
C done
B done
A done
B done
A done
A done
```

10. (5 pt.) C++ offers three kinds of inheritance; public, private and protected inheritance. Briefly describe three types of inheritance.

- **Public inheritance** – public member and protected member of the base class are inherited as a public member and protected member of the derived class.
- **Private inheritance** – public and protected member of the base class become private member of the derived class.
- **Protected inheritance** – public and protected member of the base class become protected member of the derived class.