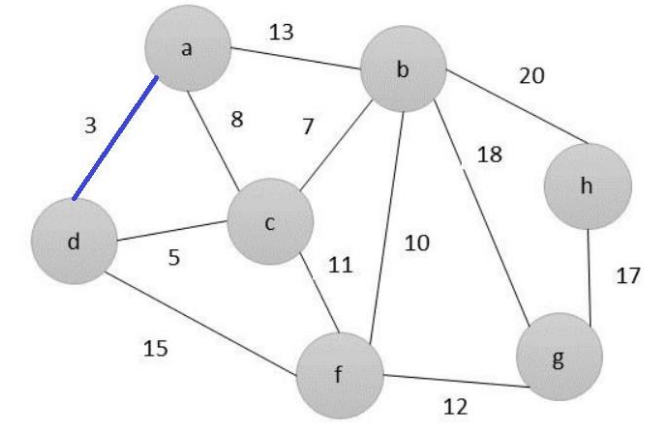
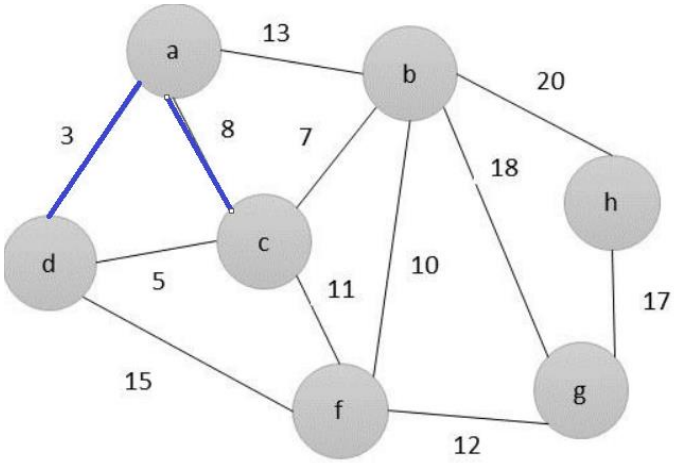
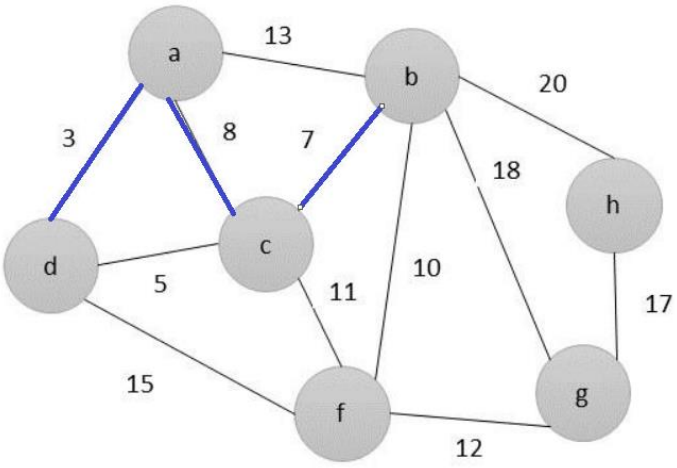
Question 1

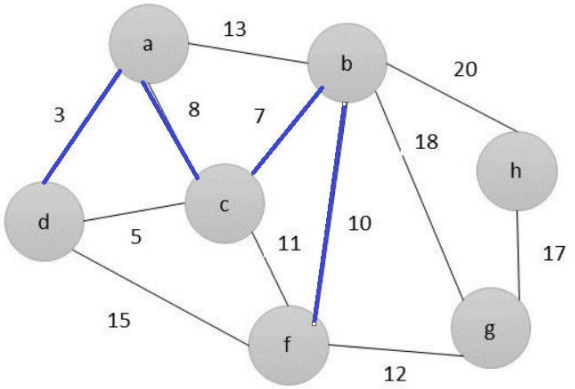
1.1)

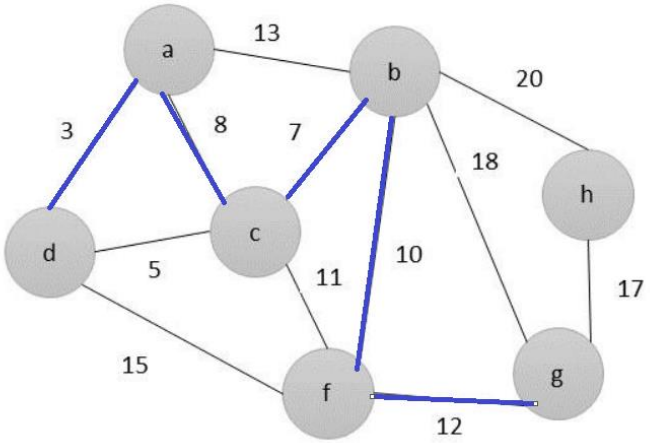
Given the graph below

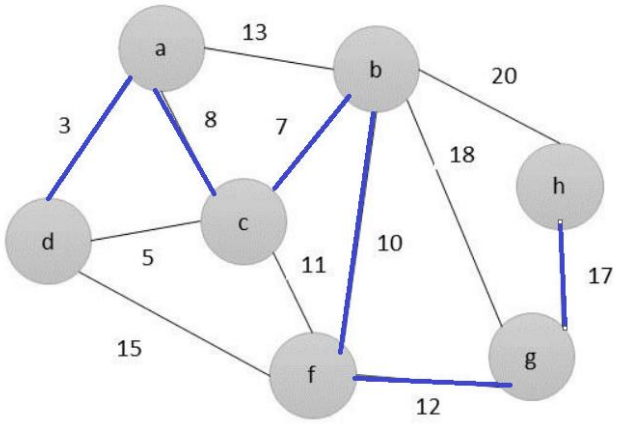












Visited { d, a , c, b, f, g, h }

1.2) Visited { d, a , c, b, f, g, h }

= 3 + 8 +7 + 10 + 12 + 17

= 57 cost of MST

1.3) Possible Vertex is F and B or A

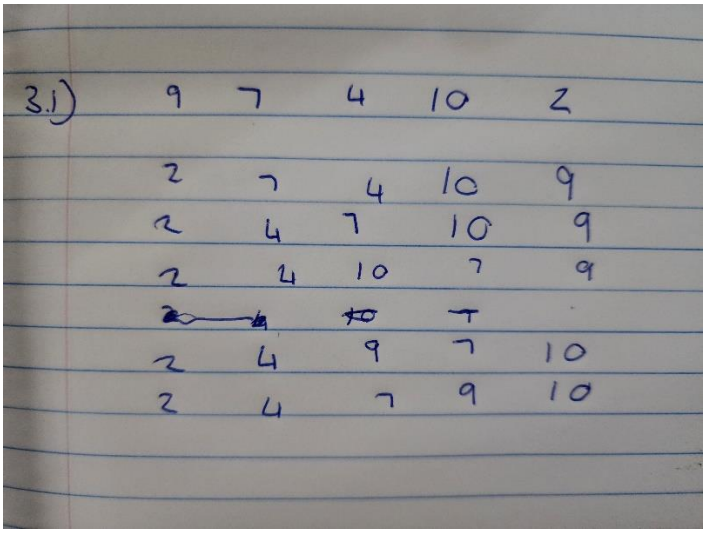
Quesiton 2

```
using System.Collections.Generic;
using System;
class Question2
{
```

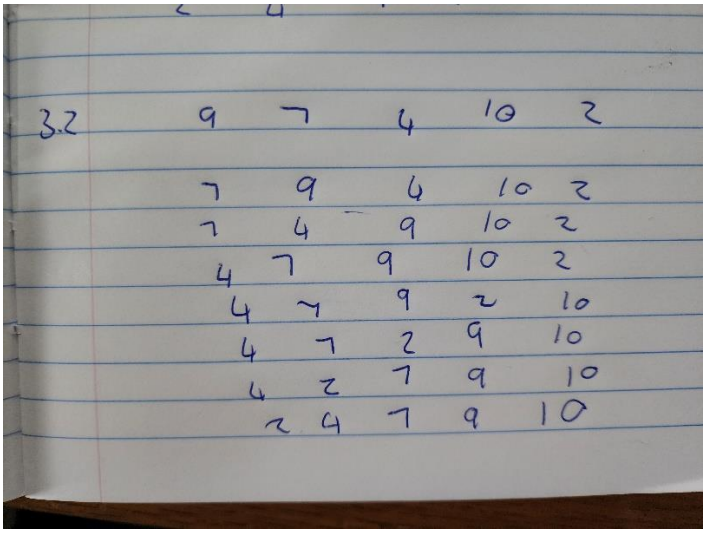
```
static void Main()
{
    List<int> numberlist = new List<int>();

    numberlist.Add(56);
    numberlist.Add(85);
    numberlist.Add(42);
    numberlist.Add(73);
    numberlist.Add(65);
    numberlist.Add(30);
    numberlist.Add(90);
    numberlist.Add(75);

    // Displaying the initial content
    Console.Write("numlist Array: \n");
    Console.Write("Initial Content: ");
    foreach (int n in numberlist)
    {
        Console.Write(n + " ");
    }
    Console.WriteLine();
    //write number of elements in array
    Console.Write("Total Number of Elements: " + numberlist.Capacity);
    Console.WriteLine();

    //find the 30 .
    int find30 = numberlist.BinarySearch(30);
    if (find30 > 0)
    {
        Console.WriteLine("False");
    }
    else
    {
        Console.WriteLine("True");
    }
    //find the 70.
    int find70 = numberlist.BinarySearch(70);
    if (find70 < 0)
    {
        Console.WriteLine("False");
    }
    else
    {
        Console.WriteLine("True");
    }
    numberlist.Remove(90);
    // Displaying the final content after removing 90
    Console.Write("Final Content: ");
    foreach (int n in numberlist)
    {
        Console.Write(n + " ");
    }
}
```

Question 3

3.1)



3.2)

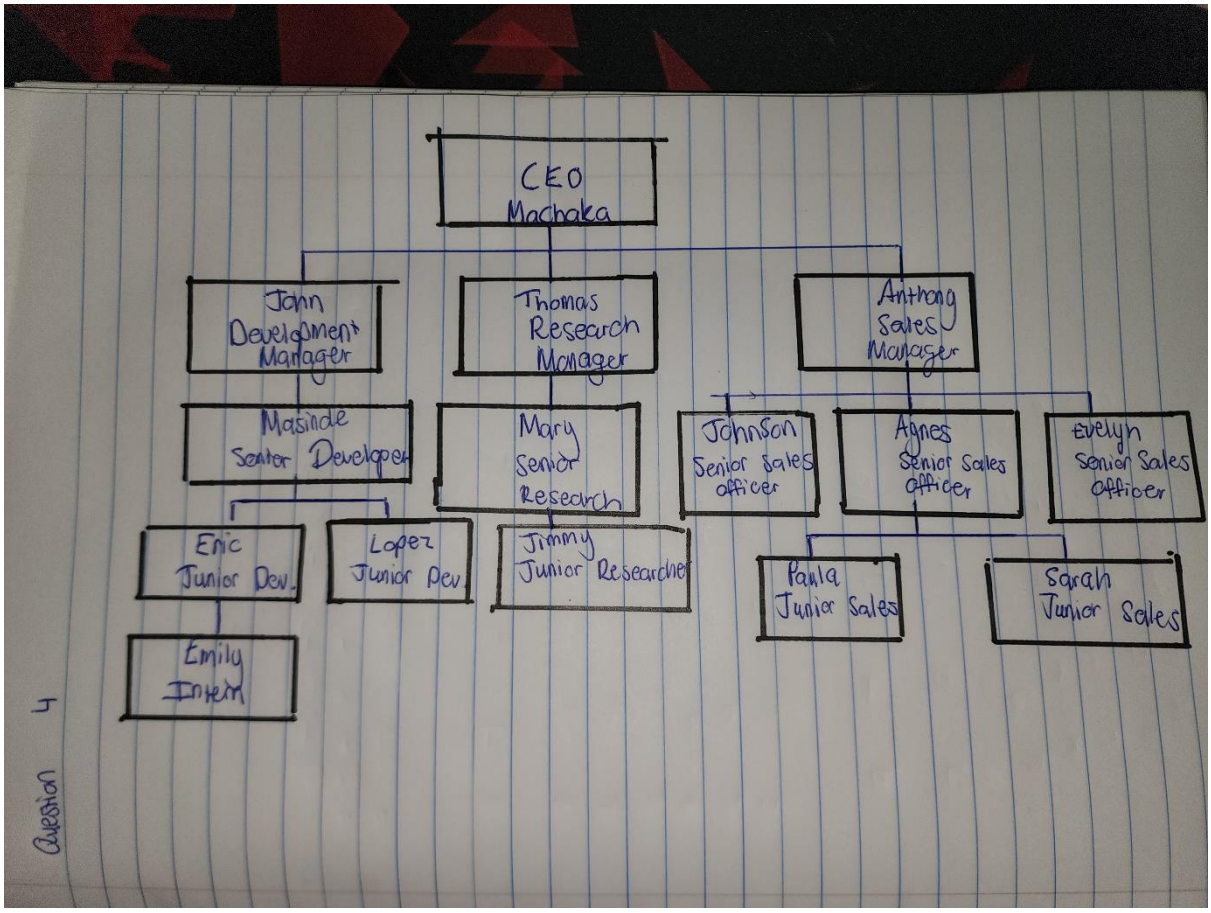


3.3) The selection sort algorithm sorts an array by repeatedly finding the minimum element from the unsorted part, while considering ascending ordering, and putting it at the beginning. In a given array, the algorithm maintains two subarrays.

Insertion sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is essentially divided between sorted and unsorted parts. Values are chosen and assigned to the appropriate positions in the sorted part of the data from the unsorted part.

Question 4

4.1



4.2 The nodes are represented by their “Department and roles” in the above organizational tree structure:

These nodes include Development Manager, Research Manager, Sales Manager;

Then Senior Developer, Senior Research, Senior Sales officed x3;

Then Junior Developer x2 with the intern, Junior Researcher and Junior Sales x2

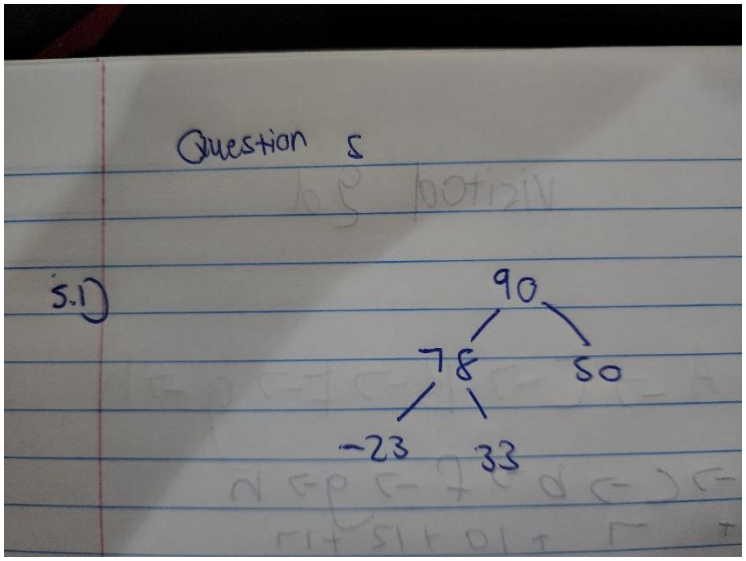
4.3

```
using System;
using System.Text;


public class Node
{
	public int data;
	public Node left, right;

	public Node(int item)
	{
		data = item;
		left = right = null;
	}
}

public class BinaryTree
{
```

```
        public Node root;

        public virtual void printPaths(Node node)
        {
                int[] path = new int[100];
                printPathsRecur(node, path, 0);
        }

        public virtual void printPathsRecur(Node node, int[] path, int pathLen)
        {
                if (node == null)
                {
                        return;
                }

                path[pathLen] = node.data;
                pathLen++;

                if (node.left == null && node.right == null)
                {
                        printArray(path, pathLen);
                }
                else
                {
                        printPathsRecur(node.left, path, pathLen);
                        printPathsRecur(node.right, path, pathLen);
                }
        }

        public virtual void printArray(int[] ints, int len)
        {
                int i;
                for (i = 0; i < len; i++)
                {
                        Console.Write(ints[i] + " "); // make space between the nodes
                }
                Console.WriteLine(" ");
        }


        public static void Main(string[] args)
        { //CODE TO PRINT THE DIFFERENT PATHS
                BinaryTree TREE = new BinaryTree();
                TREE.root = new Node(30);
                TREE.root.left = new Node(20);
                TREE.root.right = new Node(65);
                TREE.root.left.left = new Node(15);
                TREE.root.left.right = new Node(35);
                TREE.root.right.left = new Node(55);
                TREE.root.right.right = new Node(70);
                TREE.root.left.left.left = new Node(13);
                TREE.root.left.left.right = new Node(25);

                TREE.printPaths(TREE.root);
        }
}
```

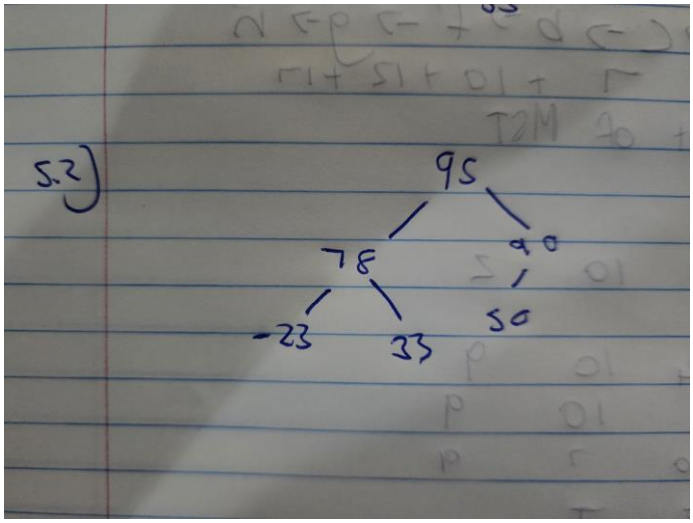
Question 5

5.1



90	78	50	-23	33

5.2



95	78	90	-23	33	50

5.3

```
using System;
namespace HeapSort
{
    public class Question4
    {
        static void heapSort(int[] arr, int n)
        {
            for (int i = n / 2 - 1; i >= 0; i--)
                heapify(arr, n, i);
            for (int i = n - 1; i >= 0; i--)
            {
                int temp = arr[0];
                arr[0] = arr[i];
                arr[i] = temp;
```

```
                heapify(arr, i, 0);
            }
        }
        static void heapify(int[] arr, int n, int i)
        {
            int largest = i;
            int left = 2 * i + 1;
            int right = 2 * i + 2;
            if (left < n && arr[left] > arr[largest])
                largest = left;
            if (right < n && arr[right] > arr[largest])
                largest = right;
            if (largest != i)
            {
                int swap = arr[i];
                arr[i] = arr[largest];
                arr[largest] = swap;
                heapify(arr, n, largest);
            }
        }
        public static void Main()
        {
            int[] arr = {50, 33, 78, -23, 90  };
            int n = 5, i;
            Console.WriteLine("Heap Sort");
            Console.Write("Initial array(hst) is: ");
            for (i = 0; i < n; i++)
            {
                Console.Write(arr[i] + " ");
            }
            heapSort(arr, 5);
            Console.Write(" Sorted Array(hst) is: ");
           for (i = 0; i < n; i++)
            {
                Console.Write(arr[i] + " ");

            }
        }

    }
}
```