**Task No. 1:** Implement Quick Sort Algorithm on integer array  **Solution:**class Program

{

public static void Sort(int[] a, int n)

{

Sort(a, 0, n - 1);

}

private static void Sort(int[] a, int low, int up)

{

if (low >= up) // zero or one element in sublist

return;

int p = Partition(a, low, up);

Sort(a, low, p - 1); // sort left sublist

Sort(a, p + 1, up); //sort right sublist

}

private static int Partition(int[] a, int low, int up)

{

int temp, i, j, pivot;

pivot = a[low];

i = low + 1; // moves from left to right

j = up; // moves from right to left

while (i <= j)

{

while (a[i] < pivot && i < up)

i++;

while (a[j] > pivot)

j--;

if (i < j) // swap a[i] and a[j]

{

temp = a[i];

a[i] = a[j];

a[j] = temp;

i++;

j--;

}

else //found proper place for pivot

break;

}

//proper place for pivot is j

a[low] = a[j];

a[j] = pivot;

return j;

}

static void Main(string[] args)

{

int i, n;

int[] a = new int[20];

Console.Write("Enter the number of elements in array: ");

n = Convert.ToInt32(Console.ReadLine());

for (i = 0; i < n; i++)

{

Console.Write("Enter Elements " + (i + 1) + ":");

a[i] = Convert.ToInt32(Console.ReadLine());

}

Sort(a, n);

Console.WriteLine("Sorted Array is: ");

for (i = 0; i < n; i++)

{

Console.Write(a[i]);

Console.WriteLine();

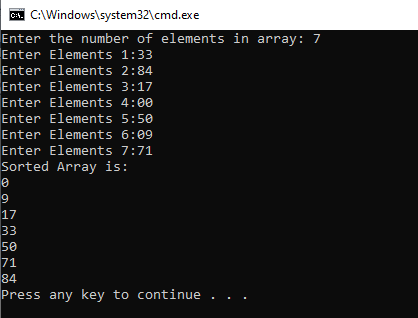
}

}

}

}

**Output:**



**Task No. 2:** Implement Quick Sort Algorithm on string array using left value as first pivoting value.

**Solution:**

class Program

{

public static void QuickSort(string[] A, int lb, int ub)

{

if (lb < ub)

{

int pv = lb;

int lo = lb + 1;

int hi = ub;

while ((lo <= hi) && (hi >= lb) && (lo <= ub))

{

if (A[pv].CompareTo(A[lo]) > 0)

{

lo++;

continue;

}

if (A[pv].CompareTo(A[hi]) < 0)

{

hi--;

continue;

}

else

{

string temp = A[lo];

A[lo] = A[hi];

A[hi] = temp;

}

}

string temp1 = A[pv];

A[pv] = A[hi];

A[hi] = temp1;

pv = hi;

QuickSort(A, lb, pv - 1);

QuickSort(A, pv + 1, ub);

}

}

static void Main(string[] args)

{

Console.WriteLine("Unsorted Array\n");

string[] A = new string[] { "jkl", "bcd", "poq", "ijk", "qqq" };

foreach (string item in A)

{

Console.WriteLine(item + " ");

}

Console.WriteLine();

int lb = 0;

int ub = A.Length - 1;

QuickSort(A, lb, ub);

Console.WriteLine("After Applying Quick Sort\n");

Console.WriteLine("\nSorted Array\n");

foreach (string item in A)

{

Console.WriteLine(item + " ");

}

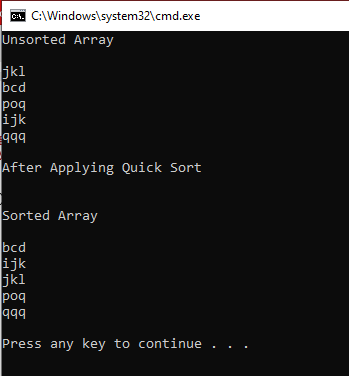
Console.WriteLine();

}

}

}

**Output:**



**Task No. 3:** Implement Quick Sort Algorithm and design Windows Form Application in which you have to sort user input values in ascending and descending order as user requires, take right value as first pivoting value and show all the pivot values in a sequence they are selected.

**Solution:**

Quick Sort:-

public static int Partition(string[] A, int LB, int UB)

{

int Pv = UB, Lo = LB , Hi = UB-1;

while (Lo <= UB && Hi >= LB)

{

if (int.Parse(A[Hi]) > int.Parse(A[Pv]))

{

Hi--; continue;

}

if (int.Parse(A[Lo]) < int.Parse(A[Pv]))

{

Lo++; continue;

}

if (Lo < Hi)

{

string temp1 = A[Hi];

A[Hi] = A[Lo];

A[Lo] = temp1;

}

else

{

break;

}

}

string temp = A[Pv];

A[Pv] = A[Lo];

A[Lo] = temp;

Pv = Lo;

return Pv;

}

public static void QuickSorting(string[] A, int LB, int UB)

{

if (LB <= UB)

{

int Pv = Partition(A, LB, UB);

PVs.Add(Pv);

QuickSorting(A, LB, Pv - 1);

QuickSorting(A, Pv + 1, UB);

}

return;

}

public static int PartitionDescending(string[] A, int LB, int UB)

{

int Pv = UB, Lo = LB, Hi = UB-1;

while (Lo <= UB && Hi >= LB)

{

if (int.Parse(A[Hi]) < int.Parse(A[Pv]))

{

Hi--; continue;

}

if (int.Parse(A[Lo]) > int.Parse(A[Pv]))

{

Lo++; continue;

}

if (Lo < Hi)

{

string temp1 = A[Hi];

A[Hi] = A[Lo];

A[Lo] = temp1;

}

else

{

break;

}

}

string temp = A[Pv];

A[Pv] = A[Lo];

A[Lo] = temp;

Pv = Lo;

return Pv;

}

public static void QuickSortingDescending(string[] A, int LB, int UB)

{

if (LB <= UB)

{

int Pv = PartitionDescending(A, LB, UB);

PVs.Add(Pv);

QuickSortingDescending(A, LB, Pv - 1);

QuickSortingDescending(A, Pv + 1, UB);

}

return;

}

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button3\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void btnclose\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void button1\_Click(object sender, EventArgs e)

{

InputTxt.Text = OutputTxt.Text = PivotTxt.Text= string.Empty;

OrderBox.Text = string.Empty;

}

public void RefreshAnswers()

{

OutputTxt.Text = PivotTxt.Text = string.Empty;

}

private void SortBtn\_Click(object sender, EventArgs e)

{

try

{

QuickSort.PVs.Clear();

RefreshAnswers();

if (OrderBox.Text == "Ascending")

{

string input = InputTxt.Text;

string[] array = input.Split(',');

QuickSort.QuickSorting(array, 0, (array.Length - 1));

for (int i = 0; i < array.Length; i++)

{

OutputTxt.AppendText(array[i] + "\n");

}

string Pivots = "";

for (int i = 0; i < QuickSort.PVs.Count; i++)

{

Pivots += array[QuickSort.PVs[i]];

if (i < QuickSort.PVs.Count - 1)

{

Pivots += ",";

}

}

PivotTxt.Text = Pivots;

}

else if (OrderBox.Text == "Descending")

{

string input = InputTxt.Text;

string[] array = input.Split(',');

QuickSort.QuickSortingDescending(array, 0, (array.Length - 1));

for (int i = 0; i < array.Length; i++)

{

OutputTxt.AppendText(array[i] + "\n");

}

string Pivots = "";

for (int i = 0; i < QuickSort.PVs.Count; i++)

{

Pivots += array[QuickSort.PVs[i]];

if (i < QuickSort.PVs.Count - 1)

{

Pivots += ",";

}

}

PivotTxt.Text = Pivots;

}

else

{

MessageBox.Show("Kindly select sorting type");

}

}

catch (Exception)

{

MessageBox.Show("An error occured");

}

}

**Output:**

