Bahria University,

Karachi Campus

A picture containing text, room

Description automatically generated

LAB EXPERIMENT NO.

\_\_\_\_**07**\_\_\_\_\_

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| 1 | Implement recursive method of merge sort algorithm to sort an array of 10 characters. |
| 2 | Implement Quick Sort Algorithm on string array using left value as first pivoting value. |
|  |  |
|  |  |
|  |  |
|  |  |

Submitted On

14/12/2022

(Date: DD/MM/YY)

**Task No. 1 :** **Implement recursive method of merge sort algorithm to sort an array of 10 characters.**

**Solution:**

**Main Class:**

char[] arr = { 'a','y','e','s','h','a','k','h','a','n'};

            Sort sort = new Sort();

            Console.Write("Before Sort = ");

            sort.print(arr);

            sort.mergeSort(arr, 0,9);

            Console.Write("\nAfter Sort = ");

            sort.print(arr);

            Console.ReadKey();

**Sort Class:**

public class Sort {

        public void mergeSort(char[] arr, int left, int right) {

            if (left < right) {

                int mid = (left + right) / 2;

                mergeSort(arr, left, mid);

                mergeSort(arr, mid + 1, right);

                merge(arr, left, mid, right); } }

        protected void merge(char[] arr, int left, int mid, int right) {

            int size1 = (mid - left) + 1;

            int size2 = right - mid;

            char[] left\_array = new char[size1];

            char[] right\_array = new char[size2];

            int i = 0, j = 0;

            while (i < size1) {

                left\_array[i] = arr[left + i];

                i++; }

            while (j < size2) {

                right\_array[j] = arr[mid + 1 + j];

                j++; }

            int k = left;

            i = 0; j = 0;

            while (i < size1 && j < size2) {

                if (left\_array[i].CompareTo(right\_array[j]) > 0) {

                    arr[k++] = right\_array[j++]; }

                else {

                    arr[k++] = left\_array[i++]; } }

            while (i < size1) {

                arr[k++] = left\_array[i++]; }

            while (j < size2) {

                arr[k++] = right\_array[j++]; } }

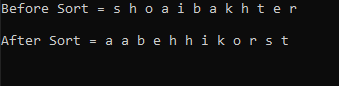
        public void print(char[] arr) {

            foreach (char item in arr) {

                Console.Write(item + " "); }

            Console.WriteLine(); } }

**Output:**



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

**Solution:**

**Main Class:**

string[] arr = { "shoaib", "akhter", "jawwad", "surfaraz", "hussain", "usama"};

            Sort obj = new Sort();

            Console.Write("Before Quick Sorting = ");

            obj.printArray(arr);

            obj.Quick(arr, 0, 5);

            Console.Write("\n\nAfter  Quick Sorting = ");

            obj.printArray(arr);

            Console.ReadKey();

**Sort Class**

internal class Sort {

        private void swap(string[] arr, int l,int r) {

            string temp  = arr[l];

            arr[l] = arr[r];

            arr[r] = temp;   }

        public void printArray(string[] arr) {

            foreach (string i in arr)

                Console.Write("{0}   ",i); }

        public void Quick(string[] arr,int left,int right) {

            if(left < right) {

                int pivot\_Position = partition(arr, left, right);

                Quick(arr, left, pivot\_Position - 1);

                Quick(arr, pivot\_Position + 1, right); }       }

        private int partition(string[] arr , int low, int high) {

            string pivot = arr[low];

            int i = low;

            int j = high;

            while (i < j) {

                while (arr.Length > i && arr[i].CompareTo(pivot) <= 0)

                    i++;

                while (arr[j].CompareTo(pivot) > 0)

                    j--;

                if (i < j)

                    swap(arr, i, j);

                else if(j < i)

                    swap(arr, j, low); }

            return j; } }

**Output:**

Graphical user interface, text

Description automatically generated