DAA LAB PROGRAM

Name: - Mohd Nasir **University Roll No: -2016855**

Section: - 'F' **Student Id: -20021595**

Roll No: - 14 **Enrolment No: -GE202016855**

WEEK 1

/*Ques1. Given an array of nonnegative integers, design a linear algorithm and implement it using a program to find whether given key element is present in the array or not. Also, find total number of comparisons for each input case. (Time Complexity = O(n), where n is the size of input)*/

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
bool LinearSearch(int *arr, int size, int *comparisons);
int main()
{
    int test;
    printf("Please Enter no of Test :");
    scanf("%d", &test);
    while (test--)
    {
        int size, comparisons = 0;
        printf("Enter Array size :");
        scanf("%d", &size);
        printf("Input Array Elements.\n");
        int *arr = (int *)malloc(size * sizeof(int));
        for (int i = 0; i < size; i++)
            scanf("%d", &arr[i]);
```

```
        if (LinearSearch(arr, size, &comparisons))
            printf("Present ");
        else
            printf("Not Present ");
        printf("%d\n", comparisons);
    }

    return 0;
}
bool LinearSearch(int *arr, int size, int *comparisons)
{
    int key;
    printf("Please Enter a key :");
    scanf("%d", &key);
    for (int i = 0; i < size; i++)
    {
        *(comparisons) = *(comparisons) + 1;
        if (arr[i] == key)
            return true;
    }
    return false;
}
```

Output Clear

```
$ Please Enter no of Test :3
Enter Array size :8
Input Array Elements.
34 35 65 31 25 89 64 30
Please Enter a key :89
Present 6
Enter Array size :5
Input Array Elements.
977 354 244 546 355
Please Enter a key :244
Present 3
Enter Array size :6
Input Array Elements.
23 64 13 67 43 56
Please Enter a key :63
Not Present 6
```

/*Ques2(Week1). Given an already sorted array of positive integers,design an algorithm and implement it using a program to find whether given key element is present in the array or not. Also, find total number of comparisons for each input case. (Time Complexity = O(nlogn), where n is the size of input).*/

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
bool BinarySearch(int *arr, int low, int high, int *comparisons);
int main()
{
    int size, comparisons = 0;
    printf("Enter Array size :");
    scanf("%d", &size);
    printf("Input Array Elements.\n");
    int *arr = (int *)malloc(size * sizeof(int));
    for (int i = 0; i < size; i++)
        scanf("%d", &arr[i]);
    if (BinarySearch(arr, 0, size, &comparisons))
        printf("Present ");
    else
        printf("Not Present ");
    printf("%d", comparisons);
    return 0;
}
bool BinarySearch(int *arr, int low, int high, int *comparisons)
{
    int key, mid;
    printf("Please Enter a key :");
    scanf("%d", &key);
```

```
    while (low < high)
    {
        mid = (low + high) / 2;
        *(comparisons) = *(comparisons) + 1;
        if (arr[mid] == key)
            return true;
        else if (arr[mid] > key)
            high = mid - 1;
        else
            low = mid + 1;
    }
    return false;
}
```

Output Clear

```
$ Enter Array size :8
Input Array Elements.
1 2 3 4 5 6 7 8
Please Enter a key :5
Present 1
Enter Array size :10
Input Array Elements.
65 99 100 101 105 110 200 2007 2009 20010
Please Enter a key :500
Not Present 3
```

/*Ques3(WEEK1). Given an already sorted array of positive integers, design an algorithm and implement it using a program to find whether a given key element is present in the sorted array or not. For an array arr[n], search at the indexes arr[0], arr[2], arr[4], ...,arr[2k] and so on. Once the interval (arr[2k] < key < arr [2k+1]) is found, perform a linear search operation from the index 2k to find the element key. (Complexity < O(n), where n is the number of elements need to be scanned for searching): */

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include <math.h>
bool BinarySearch(int *arr, int size, int *comparisons);
int main()
{
    int test;
    printf("Please Enter No of Test :");
    scanf("%d", &test);
    while (test--)
    {
        int size, comparisons = 0;
        printf("Enter Array size :");
        scanf("%d", &size);
        printf("Input Array Elements.\n");
        int *arr = (int *)malloc(size * sizeof(int));
        for (int i = 0; i < size; i++)
            scanf("%d", &arr[i]);
        if (BinarySearch(arr, size, &comparisons))
            printf("Present\n");
        else
            printf("Not Present\n");
        printf("%d", comparisons);
```

```
    }
    return 0;
}
bool BinarySearch(int *arr, int size, int *comparisons)
{
    int key, mid, j;
    printf("Please Enter a key :");
    scanf("%d", &key);
    for (int i = 0; j < size; i++)
    {
        j = pow(2, i);
        *(comparisons) = *(comparisons) + 1;
        if (arr[j] == key)
            return true;
    }
    return false;
}
```

Output Clear

```
$ Please Enter No of Test :3
Enter Array size :5
Input Array Elements.
12 23 36 39 41
Please Enter a key :41
Present
3
Enter Array size :8
Input Array Elements.
21 39 40 45 51 54 68 72
Please Enter a key :69
Not Present
4
Enter Array size :10
Input Array Elements.
101 246 438 561 796 896 899 4644 7999 8545
Please Enter a key :7999
Present
4
```

WEEK 2

/*Ques1(Week2). Given a sorted array of positive integers containing few duplicate elements, design an algorithm and implement it using a program to find whether the given key element is present in the array or not. If present, then also find the number of copies of given key. (Time Complexity = O(log n))*/

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
int BinarySearch(int *arr, int low, int high, int *comparisons);
int main()
{
   int test;
   printf("Input No of Test :");
   scanf("%d", &test);
   while (test--)
   {
      int size, comparisons = 0, index = 0;
      printf("Enter Array size :");
      scanf("%d", &size);
      printf("Input Array Elements.\n");
      int *arr = (int *)malloc(size * sizeof(int));
      for (int i = 0; i < size; i++)
         scanf("%d", &arr[i]);
      index = BinarySearch(arr, 0, size, &comparisons);
      int i = index - 1, j = index + 1, count = 1;
      if (!index)
         printf("Not Present ");
      else
      {
```

```
        while (arr[i] == arr[index] || arr[j] == arr[index])
        {
            if ((i < 0 && j >= size) || (arr[i] != arr[index] && arr[j] != arr[index]))
                break;
            if (i >= 0 && arr[i] == arr[index])
            {
                i--;
                count++;
            }
            if (j < size && arr[j] == arr[index])
            {
                j++;
                count++;
            }
        }
        printf("%d -> %d\n", arr[index], count);
      }
   }


   return 0;
}
int BinarySearch(int *arr, int low, int high, int *comparisons)
{
   int key, mid;
   printf("Please Enter a key :");
   scanf("%d", &key);
   while (low < high)
   {
      mid = (low + high) / 2;
      *(comparisons) = *(comparisons) + 1;
      if (arr[mid] == key)
```

```
            return mid;
        else if (arr[mid] > key)
            high = mid - 1;
        else
            low = mid + 1;
    }
    return 0;
}
```

Output Clear

```
$ Input No of Test :2
Enter Array size :10
Input Array Elements.
235 235 278 278 763 764 790 853 981 981
Please Enter a key :981
981 -> 2
Enter Array size :15
Input Array Elements.
1 2 2 3 3 5 5 5 25 75 75 75 97 97 97
Please Enter a key :75
75 -> 3
```

/*Ques2(Week2).Given a sorted array of positive integers, design an algorithm and implement it using a program to find three indices i, j, k such that arr[i] + arr[j] = arr[k]*/

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
void IndexFound(int *arr, int size);
int main()
{
   int test;
   printf("Please Input No of Test Caase :");
   scanf("%d", &test);
   while (test--)
   {
      int size;
      printf("Enter Array size :");
      scanf("%d", &size);
      printf("Input Array Elements.\n");
      int *arr = (int *)malloc(size * sizeof(int));
      for (int i = 0; i < size; i++)
         scanf("%d", &arr[i]);
      IndexFound(arr, size);
   }

   return 0;
}
void IndexFound(int *arr, int size)
{
   int i, j, k;
   for (i = 0; i < size; i++)
```

```
        for (j = i + 1; j < size; j++)
            for (k = j + 1; k < size; k++)
                if ((arr[i] + arr[j]) == arr[k])
                {
                    printf("%d, %d, %d\n", i + 1, j + 1, k + 1);
                    break;
                }
    if (i == size)
        printf("No Sequence Found\n");
}
```

Output Clear

```
$ Please Input No of Test Caase :3
Enter Array size :5
No Sequence Found.
Input Array Elements.
1 5 84 209 341
Enter Array size :10
Input Array Elements.
24 28 48 71 86 89 92 120 194 201
2, 7, 8
Enter Array size :15
Input Array Elements.
64 69 82 95 99 107 113 141 171 350 369 400 511 590 666
1, 6, 9
```

/*Ques3(Week2).Given an array of nonnegative integers, design an algorithm and a program to count the number of pairs of integers such that their difference is equal to a given key, K.*/

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
int DiffrencePairs(int *arr, int size);
int main()
{
   int test;
   printf("Please input no of test case :");
   scanf("%d", &test);
   while (test--)
   {
      int size;
      printf("Enter Array size :");
      scanf("%d", &size);
      printf("Input Array Elements.\n");
      int *arr = (int *)malloc(size * sizeof(int));
      for (int i = 0; i < size; i++)
         scanf("%d", &arr[i]);
      printf("%d\n", DiffrencePairs(arr, size));
   }

   return 0;
}
int DiffrencePairs(int *arr, int size)
{
   int diff, count = 0;
   printf("Please Enter a Difference :");
```

```
    scanf("%d", &diff);
    for (int i = 0; i < size; i++)
        for (int j = i + 1; j < size; j++)
            if (abs(arr[i] - arr[j]) == diff)
                count++;
    return count;
}
```

Output Clear

```
$ Please input no of test case :2
Enter Array size :5
Input Array Elements.
1 51 84 21 31
Please Enter a Difference :20
2
Enter Array size :10
Input Array Elements.
24 71 16 92 12 28 48 14 20 22
Please Enter a Difference :4
4
```

WEEK 3

/*Ques1(Week3).Given an unsorted array of integers, design an algorithm and a program to sort the array using insertion sort. Your program should be able to find number of comparisons and shifts (shifts total number of times the array elements are shifted from their place) required for sorting the array.*/

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
void InsertionSort(int *arr, int size, int *shifts, int *comparisions);
void PrintArray(int *arr, int size);
int main()
{
   int test;
   printf("Please Input No of Test Case :");
   scanf("%d", &test);
   while (test--)
   {
      int size, shifts = 0, comparisions = 0;
      printf("Enter Array size :");
      scanf("%d", &size);
      printf("Input Array Elements.\n");
      int *arr = (int *)malloc(size * sizeof(int));
      for (int i = 0; i < size; i++)
         scanf("%d", &arr[i]);
      InsertionSort(arr, size, &shifts, &comparisions);
      PrintArray(arr, size);
      printf("\nComparisions = %d\nShifts = %d\n", comparisions, shifts);
   }
```

```
    return 0;
}
void InsertionSort(int *arr, int size, int *shifts, int *comparisions)
{
    int i, j, key;
    for (i = 1; i < size; i++)
    {
        key = arr[i];
        j = i - 1;
        while (j >= 0 && key < arr[j])
        {
            *(comparisions) = *(comparisions) + 1;
            arr[j + 1] = arr[j];
            j--;
        }
        *(shifts) = *(shifts) + 1;
        arr[j + 1] = key;
    }
    *(shifts) = *(shifts) + *(comparisions);
}
void PrintArray(int *arr, int size)
{
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", arr[i]);
}
```

Output Clear

```
$ Please Input No of Test Case :3
Enter Array size :8
Input Array Elements.
-23 65 -31 76 46 89 45 32
-31 -23 32 45 46 65 76 89
Comparisions = 13
Shifts = 20
Enter Array size :10
Input Array Elements.
54 65 34 76 78 97 46 32 51 21
21 32 34 46 51 54 65 76 78 97
Comparisions = 28
Shifts = 37
Enter Array size :15
Input Array Elements.
63 42 223 645 652 31 324 22 553 -12 54 65 86 46 325
-12 22 31 42 46 54 63 65 86 223 324 325 553 645 652
Comparisions = 54
Shifts = 68
```

/*Ques2(Week3) .Given an unsorted array of integers, design an algorithm and implement a program to sort this array using selection sort. Your program should also find number of comparisons and number of swaps required*/

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
void SelectionSort(int *arr, int size, int *swap, int *comparisions);
void PrintArray(int *arr, int size);
int main()
{
   int test;
   printf("Please Input no of Test :");
   scanf("%d", &test);
   while (test--)
   {
      int size, swap = 0, comparisions = 0;
      printf("Enter Array size :");
      scanf("%d", &size);
      printf("Input Array Elements.\n");
      int *arr = (int *)malloc(size * sizeof(int));
      for (int i = 0; i < size; i++)
         scanf("%d", &arr[i]);
      SelectionSort(arr, size, &swap, &comparisions);
      PrintArray(arr, size);
      printf("\nComparisions = %d\nSwaps = %d\n", comparisions, swap);
   }

   return 0;
}
void SelectionSort(int *arr, int size, int *swap, int *comparisions)
{
```

```
    int min, pos, i, j;
    for (i = 0; i < size - 1; i++)
    {
        min = arr[i];
        pos = i;
        for (j = i + 1; j < size; j++)
        {
            *(comparisions) = *(comparisions) + 1;
            if (min > arr[j])
            {
                min = arr[j];
                pos = j;
            }
        }
        /**(swap) = *(swap) + 1;*/
        if (pos != i)
        {
            *(swap) = *(swap) + 1;
            arr[pos] = arr[i];
            arr[i] = min;
        }
    }
    printf("%d %d\n", comparisions, swap);
}
void PrintArray(int *arr, int size)
{
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", arr[i]);
}
```

Output Clear

```
$ Please Input no of Test :3
Enter Array size :8
Input Array Elements.
-13 65 -21 76 46 89 45 12
6618624 6618628
-21 -13 12 45 46 65 76 89
Comparisions = 28
Swaps = 7
Enter Array size :10
Input Array Elements.
54 65 34 76 78 97 46 32 51 21
6618624 6618628
21 32 34 46 51 54 65 76 78 97
Comparisions = 45
Swaps = 9
Enter Array size :15
Input Array Elements.
63 42 223 645 652 31 324 22 553 12 54 65 86 46 325
12 22 31 42 46 54 63 65 86 223 324 325 553 645 652
Comparisions = 105
Swaps = 14
```

/*Ques3(Week3). Given an unsorted array of positive integers, design an algorithm and implement it using a program to find whether there are any duplicate elements in the array or not. (use sorting) (Time Complexity = O(n log n))*/

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
bool Dublicates(int *arr, int size);
int MaximumElement(int *arr, int size);
int main()
{
   int Test;
   printf("Please Input no of Test Case :");
   scanf("%d", &Test);
   while (Test--)
   {
      int size;
      printf("Enter Array size :");
      scanf("%d", &size);
      printf("Input Array Elements.\n");
      int *arr = (int *)malloc(size * sizeof(int));
      for (int i = 0; i < size; i++)
         scanf("%d", &arr[i]);
      if (Dublicates(arr, size))
         printf("YES\n");
      else
         printf("NO\n");
   }

   return 0;
}
bool Dublicates(int *arr, int size)
```

```
{
    int *temp = (int *)malloc((MaximumElement(arr, size) + 1) * sizeof(int));
    int i = 0;
    for (i = 0; i <= (MaximumElement(arr, size)); i++)
        temp[i] = 0;
    for (i = 0; i < size; i++)
        temp[arr[i]] = temp[arr[i]] + 1;
    for (i = 0; i <= (MaximumElement(arr, size)); i++)
        if (temp[i] >= 2)
            return true;
    return false;
}
int MaximumElement(int *arr, int size)
{
    int max = INT_MIN;
    for (int i = 0; i < size; i++)
        if (max < arr[i])
            max = arr[i];
    return max;
}
```

Output Clear

```
$ Please Input no of Test Case :3
Enter Array size :5
Input Array Elements.
28 52 83 14 75
NO
Enter Array size :10
Input Array Elements.
75 65 1 65 2 6 86 2 75 8
YES
Enter Array size :15
Input Array Elements.
75 35 86 57 98 23 73 1 64 8 11 90 61 19 20
NO
```

WEEK 4

/*Ques1(Week4).Given an unsorted array of integers, design an algorithm and implement it using a program to sort an array of elements by dividing the array into two subarrays and combining these subarrays after sorting each one of them. Your program should also find number of comparisons and inversions during sorting the array*/

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
void MergeSort(int *arr, int *temp, int low, int high);
void MergeElements(int *arr, int *temp, int low, int mid, int high);
void PrintArray(int *arr, int size);
int comparisions = 0;
int main()
{
   int test;
   printf("Please Input No of Test :");
   scanf("%d", &test);
   while (test--)
   {
      comparisions = 0;
      int size;
      printf("Enter Array size :");
      scanf("%d", &size);
      printf("Input Array Elements.\n");
      int *arr = (int *)malloc(size * sizeof(int));
      int *temp = (int *)malloc(size * sizeof(int));
      for (int i = 0; i < size; i++)
         scanf("%d", &arr[i]);
      MergeSort(arr, temp, 0, size - 1);
```

```
        PrintArray(arr, size);
        printf("\nComparisions = %d\n", comparisions);
    }

    return 0;
}
void MergeSort(int *arr, int *temp, int low, int high)
{
    int mid;
    if (low < high)
    {
        mid = (low + high) / 2;
        MergeSort(arr, temp, low, mid);
        MergeSort(arr, temp, mid + 1, high);
        MergeElements(arr, temp, low, mid, high);
    }
}
void MergeElements(int *arr, int *temp, int low, int mid, int high)
{
    int i = low, j = mid + 1, k = low;
    while (i <= mid && j <= high)
    {
        comparisions++;
        if (arr[i] <= arr[j])
        {
            temp[k] = arr[i];
            i++;
            k++;
        }
        else
        {
```

```
            temp[k] = arr[j];
            j++;
            k++;
        }
    }
    while (i <= mid)
    {
        temp[k] = arr[i];
        k++;
        i++;
    }
    while (j <= high)
    {
        temp[k] = arr[j];
        j++;
        k++;
    }
    for (i = low; i <= high; i++)
        arr[i] = temp[i];
}
void PrintArray(int *arr, int size)
{
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", arr[i]);
}
```

Output Clear

```
$ Please Input No of Test :3
Enter Array size :8
Input Array Elements.
23 65 21 76 46 89 45 32
21 23 32 45 46 65 76 89
Comparisions = 16
Enter Array size :10
Input Array Elements.
54 65 34 76 78 97 46 32 51 21
21 32 34 46 51 54 65 76 78 97
Comparisions = 22
Enter Array size :15
Input Array Elements.
63 42 223 645 652 31 324 22 553 12 54 65 86 46 325
12 22 31 42 46 54 63 65 86 223 324 325 553 645 652
Comparisions = 43
```

/*Ques2(Week4). Given an unsorted array of integers, design an algorithm and implement it using a program to sort an array of elements by partitioning the array into two subarrays based on a pivot element such that one of the sub array holds values smaller than the pivot element while another sub array holds values greater than the pivot element. Pivot element should be selected randomly from the array. Your program should also find number of comparisons and swaps required for sorting the array.*/

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
void QuickSort(int *arr, int low, int high);
int ArrayPartition(int *arr, int low, int high);
void swap(int *arr, int start, int end);
void PrintArray(int *arr, int size);
int comparisions = 0, swaps = 0;
int main()
{
   int test;
   printf("Please input no of test case :");
   scanf("%d", &test);
   while (test--)
   {
      int size;
      comparisions = 0, swaps = 0;
      printf("Enter Array size :");
      scanf("%d", &size);
      printf("Input Array Elements.\n");
      int *arr = (int *)malloc(size * sizeof(int));
      int *temp = (int *)malloc(size * sizeof(int));
```

```
        for (int i = 0; i < size; i++)
            scanf("%d", &arr[i]);
        QuickSort(arr, 0, size - 1);
        PrintArray(arr, size);
        printf("\nComparision = %d\nSwaps = %d\n", comparisions, swaps);
    }

    return 0;
}
int ArrayPartition(int *arr, int low, int high)
{
    int pivot = arr[low];
    int i = low + 1, j = high;
    while (i <= j)
    {
        while (arr[i] <= pivot)
        {
            i++;
            comparisions++;
        }
        while (arr[j] > pivot)
        {
            j--;
            comparisions++;
        }
        /*swaps++;*/
        if (i < j)
        {
            swaps++;
            int temp = arr[i];
            arr[i] = arr[j];
```

```
            arr[j] = temp;
        }
    }
    swaps++;
    arr[low] = arr[j];
    arr[j] = pivot;
    return j;
}
void swap(int *arr, int start, int end)
{
    int temp = arr[start];
    arr[start] = arr[end];
    arr[end] = temp;
}
void QuickSort(int *arr, int low, int high)
{
    int PartitionIndex;
    if (low < high)
    {
        int randomly = (rand() % (high - low + 1)) + low;
        swap(arr, low, randomly);
        PartitionIndex = ArrayPartition(arr, low, high);
        QuickSort(arr, low, PartitionIndex - 1);
        QuickSort(arr, PartitionIndex + 1, high);
    }
}
void PrintArray(int *arr, int size)
{
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", arr[i]);
```

}

Output Clear

```
$ Please input no of test case :3
Enter Array size :8
Input Array Elements.
23 65 21 76 46 89 45 32
21 23 32 45 46 65 76 89
Comparision = 16
Swaps = 8
Enter Array size :10
Input Array Elements.
54 65 34 76 78 97 46 32 51 21
21 32 34 46 51 54 65 76 78 97
Comparision = 23
Swaps = 10
Enter Array size :15
Input Array Elements.
63 42 223 645 652 31 324 22 553 12 54 65 86 46 325
12 22 31 42 46 54 63 65 86 223 324 325 553 645 652
Comparision = 43
Swaps = 16
```

/*Ques3(Week4). Given an unsorted array of integers, design an algorithm and implement it using a program to find Kth smallest or largest element in the array. (Worst case Time Complexity = O(n))*/

```
#include <stdio.h>
#include <stdlib.h>
int Partition(int *arr, int low, int high);
int QuickSelect(int *arr, int low, int high, int K);
int main()
{
   int test;
   printf("Please input no of Test :");
   scanf("%d", &test);
   while (test--)
   {
      int size, K, i;
      printf("Input the size of Array :");
      scanf("%d", &size);
      int *arr = (int *)malloc(size * sizeof(int));
      printf("Input the Array Elements.\n");
      for (i = 0; i < size; i++)
         scanf("%d", &arr[i]);
      printf("Enter the number 'K' to find 'Kth' Largest and Smallest Elements :");
      scanf("%d", &K);
      printf("%d Smallest Elemenets is :%d\n", K, QuickSelect(arr, 0, size - 1, K));
      printf("%d Largest Elemenets is :%d\n", K, QuickSelect(arr, 0, size - 1, size - K + 1));
   }

   return 0;
}
int QuickSelect(int *arr, int low, int high, int K)
{
```

```
    if (low == high)
        return arr[low];
    int PivotIndex = Partition(arr, low, high);
    int SizeofArray = PivotIndex - low + 1;
    if (SizeofArray > K)
        return QuickSelect(arr, low, PivotIndex - 1, K);
    else if (SizeofArray < K)
        return QuickSelect(arr, PivotIndex + 1, high, K - SizeofArray);
    else
        return arr[PivotIndex];
}
int Partition(int *arr, int low, int high)
{
    int pivot = arr[low];
    int i = low + 1;
    int j = high;
    while (i <= j)
    {
        while (arr[i] <= pivot)
            i++;
        while (arr[j] > pivot)
            j--;
        if (i < j)
        {
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
    arr[low] = arr[j];
    arr[j] = pivot;
```

```
    return j;
}
```

Output Clear

```
$ Please input no of Test :3
Input the size of Array :10
Input the Array Elements.
123 656 54 765 344 514 765 34 765 234
Enter the number 'K' to find 'Kth' Largest and Smallest Elements :3
3 Smallest Elemenets is :123
3 Largest Elemenets is :765
Input the size of Array :15
Input the Array Elements.
43 64 13 78 864 346 786 456 21 19 8 434 76 270 601
Enter the number 'K' to find 'Kth' Largest and Smallest Elements :8
8 Smallest Elemenets is :78
8 Largest Elemenets is :78
Input the size of Array :10
Input the Array Elements.
1 2 3 4 5 6 7 8 9 10
Enter the number 'K' to find 'Kth' Largest and Smallest Elements :8
8 Smallest Elemenets is :8
8 Largest Elemenets is :3
```

WEEK 5

/*Ques1(Week5).Given an unsorted array of alphabets containing duplicate elements. Design an algorithm and implement it using a program to find which alphabet has maximum number of occurrences and print it. (Time Complexity = O(n)) (Hint: Use counting sort)*/

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   int test;
   printf("Test Cases :");
   scanf("%d", &test);
   while (test--)
   {
      int size, i, key, max = INT_MIN, pos = 0;
      char ch;
      printf("Please Input  Array size :");
      scanf("%d", &size);
      int *arr = (int *)malloc(size * sizeof(int));
      int *temp = (int *)malloc(26 * sizeof(int));
      for (i = 0; i < 26; i++)
         temp[i] = 0;
      for (i = 0; i < size; i++)
      {
         scanf(" %c", &ch);
         key = ch - 'a';
         arr[i] = key;
         if (max < arr[i])
            max = arr[i];
      }
```

```
    for (i = 0; i < size; i++)
        temp[arr[i]] = temp[arr[i]] + 1;
    int max1 = INT_MIN;
    for (i = 0; i <= max; i++)
    {
        if (max1 < temp[i])
        {
            max1 = temp[i];
            pos = i;
        }
    }
    if (max1 >= 2)
    {
        ch = 'a' + pos;
        printf("%c -> %d\n", ch, max1);
    }
    else
        printf("No Dublicates Present.\n");
  }

  return 0;
}
```

Output Clear

```
Test Cases :3
Please Input  Array size :10
a e d w a d q a f p
a -> 3
Please Input  Array size :15
r k p g v y u m q a d j c z e
No Dublicates Present.
Please Input  Array size :20
g t ll t c w a w g l c w d s a a v c l
l -> 4
```

/*Ques2(Week5). Given an unsorted array of integers, design an algorithm and implement it using a program to find whether two elements exist such that their sum is equal to the given key element. (Time Complexity = O(n log n))*/

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
void MergeSort(int *arr, int *temp, int low, int high);
void MergeElements(int *arr, int *temp, int low, int mid, int high);
bool SumCheck(int *arr, int low, int high);
int main()
{
   int test;
   printf("Test Cases :");
   scanf("%d", &test);
   while (test--)
   {
      int size;
      printf("Enter Array size :");
      scanf("%d", &size);
      printf("Input Array Elements.\n");
      int *arr = (int *)malloc(size * sizeof(int));
      int *temp = (int *)malloc(size * sizeof(int));
      for (int i = 0; i < size; i++)
         scanf("%d", &arr[i]);
      MergeSort(arr, temp, 0, size - 1);
      if (!(SumCheck(arr, 0, size - 1)))
         printf("No Such Pairs Exist.\n");
   }

   return 0;
```

```
}
bool SumCheck(int *arr, int low, int high)
{
   int sum, temp = 0;
   printf("input sum :");
   scanf("%d", &sum);
   while (low < high)
   {
      if ((arr[low] + arr[high]) == sum)
      {
         printf("%d %d\n", arr[low], arr[high]);
         low++;
         high--;
         temp = 1;
      }
      else if ((arr[low] + arr[high]) > sum)
         high--;
      else
         low++;
   }
   if (!temp)
      return false;
   else
      return true;
}
void MergeSort(int *arr, int *temp, int low, int high)
{
   int mid;
   if (low < high)
   {
      mid = (low + high) / 2;
```

```
        MergeSort(arr, temp, low, mid);
        MergeSort(arr, temp, mid + 1, high);
        MergeElements(arr, temp, low, mid, high);
    }
}
void MergeElements(int *arr, int *temp, int low, int mid, int high)
{
    int i = low, j = mid + 1, k = low;
    while (i <= mid && j <= high)
    {
        if (arr[i] < arr[j])
        {
            temp[k] = arr[i];
            i++;
            k++;
        }
        else
        {
            temp[k] = arr[j];
            j++;
            k++;
        }
    }
    while (i <= mid)
    {
        temp[k] = arr[i];
        k++;
        i++;
    }
    while (j <= high)
    {
```

```
        temp[k] = arr[j];
        j++;
        k++;
    }
    for (i = low; i <= high; i++)
        arr[i] = temp[i];
}
```

Output Clear

```
$ Test Cases :2
Enter Array size :10
Input Array Elements.
64 28 97 40 12 72 84 24 38 10
input sum :50
10 40
12 38
Enter Array size :15
Input Array Elements.
56 10 72 91 29 3 41 45 61 20 11 39 9 12 94
input sum :302
No Such Pairs Exist.
```

/*Ques3(Week5).You have been given two sorted integer arrays of size m and n.Design an algorithm and implement it using a program to find list of elements which are common to both. (Time Complexity = O(m+n))*/

```
#include <stdio.h>
#include <stdlib.h>
void SameElements(int *arr1, int *arr2, int size1, int size2);
int main()
{
   int test;
   printf("Test Cases :");
   scanf("%d", &test);
   while (test--)
   {
      int size1, size2, i;
      printf("Enter array1 size :");
      scanf("%d", &size1);
      printf("Enter array2 size :");
      scanf("%d", &size2);
      int *arr1 = (int *)malloc(size1 * sizeof(int));
      int *arr2 = (int *)malloc(size2 * sizeof(int));
      printf("Input array1 Elemenst.\n");
      for (i = 0; i < size1; i++)
         scanf("%d", &arr1[i]);
      printf("Input array2 Elemenst.\n");
      for (i = 0; i < size2; i++)
         scanf("%d", &arr2[i]);
      SameElements(arr1, arr2, size1, size2);
      printf("\n");
   }
   return 0;
}
```

```
void SameElements(int *arr1, int *arr2, int size1, int size2)
{
   int i = 0, j = 0;
   while (i <= size1 && j <= size2)
   {
      if (arr1[i] < arr2[j])
         i++;
      else if (arr1[i] > arr2[j])
         j++;
      else
      {
         printf("%d ", arr1[i]);
         i++;
         j++;
      }
   }
}
```