MINI PROJECT

REPORT USING LATEX

0.1 Objective of the Project-

To create a program that gives useful information regarding the array of integers.

0.2 Function Description-

[**F.1**] *ascending_sorting*− To sort an given array elements in the increment order of their value. Minimum to Maximum.

[**F.2**] *descending_sorting*− To sort an given array elements in the decrement order of their value.Maximum to Minimum.

[**F.3**] *no_of_even*−To count the number of even numbers present in the array.

[**F.4**] *no_of_odd*−To count the number of odd elements present in the array.

[**F.5**] *no_of_positive_elements*− To count the number of positive numbers present in the array.

[**F.6**] *no_of_negative_elements*−To count the number of negative elements present in the array.

[**F.7**] *Sum_of_elements*−To calculate the addition of all elements present in the Array.

[**F.8**] $Mean\_of\_elements$−To calculate the average value of the given array elements.

[**F.9**] $frequency\_of\_each\_element$−To find the number of times an array elements appears in the array list.

[**F.10**] $Elements\_with\_frequency\_greater$−To find the number of elements that appeared in the array list more than k times.

[**F.11**] $elements\_with\_frequency\_lower$− To find the number of elements that appeared in the array list less than k times.

[**F.12**] $no\_of\_elements\_greater\_than\_k$− To calculate the number of elements having vaue more than k.

[**F.13**] $no\_of\_elements\_lower\_than\_k$−To find the number of elements having value less than k.

[**F.14**] $maximum\_value$−To calculate the maximum value present in the given array.

[**F.15**] $minimum\_value$−To find the minimum value element present in the array list.

0.3 Program Code-

```
#include<stdio.h>
#include<math.h>

void swap(int* a, int* b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

int ascending_sort(int arr[],int l)
{
    printf("Asceending Sorting:\n");
    int i, j, min_index;

     for (i = 0; i < l-1; i++)
     {

        // Find the minimum element in unsorted array
        min_index = i;
        for (j = i + 1; j < l; j++)
        { if (arr[j] < arr[min_index])
                min_index = j;
        }
         // Swap the found minimum element with the first element
     swap(&arr[min_index], &arr[i]);

     }
        for (i = 0; i < l; i++)
        {      printf("%d ", arr[i]);
              printf("\n");
        }
         return arr;
}
int descending_sort(int arr[],int l)
{
    printf("\nDescending Sorting:\n");
    int i,j,t=0;
```

```
  // iterates the array elements
    for (i = 0; i < l; i++) {

        // iterates the array elements from index 1
        for (j = i + 1; j < l; j++) {

            // comparing the array elements, to set array
            // elements in descending order
            if (arr[i] < arr[j]) {
                t = arr[i];
                arr[i] = arr[j];
                arr[j] = t;
            }
        }
    }
    // printing the output
    for (i = 0; i < l; i++) {
        printf("%d ", arr[i]);
        printf("\n");
    }
    return arr;
}
void no_of_even(int arr[],int l)
{
    printf("\nNumber of even elements:\n");
 int count=0;
 for(int i=0;i<l;i++)
 {
    if(arr[i]%2==0)
    count=count+1;
 }
 printf("%d\n\n",count);
}
void no_of_odd(int arr[],int l)
{
 printf("Number of odd elements:\n");
    int count=0;
 for(int i=0;i<l;i++)
 {
    if(arr[i]%2!=0)
```

```
    count=count+1;
 }
 printf("%d\n\n",count);
}
float sum_of_elements(int arr[],int l)
{
    printf("\nSum of elements:\n");
    int sum=0;
    for(int i=0;i<l;i++)
    {
        sum=sum+arr[i];
    }
    return sum;
}
float mean_of_elements(int arr[],int l,float addition)
{
    printf("\nMean of elements:\n");
    float mean=addition/l;
    return mean;
}
void frequency_of_each_element(int arr[],int l)
{
    printf("\n\n Frequency of each element:");
    int freq[l];
    int  seen=-1;
    for(int i = 0; i < l; i++)
    {
        int count = 1;
        for(int j = i+1; j < l; j++)
        {
            if(arr[i] == arr[j])
            {
                count++;
                //To avoid counting same element again
                freq[j] = seen;
            }
        }
        if(freq[i] !=seen)
            freq[i] = count;
    }
```

```
    //Displays the frequency of each element present in array
    printf("\n-------------------------\n");
    printf(" Element | Frequency\n");
    printf("-------------------------\n");
    for(int i = 0; i < l; i++)
    {
        if(freq[i] !=seen)
        {
            printf("%d", arr[i]);
            printf("\t|");
            printf(" \t%d\n", freq[i]);
        }

    }
        printf("-------------------------\n\n");

}

int elements_with_frequecncy_greater(int arr[],int l)

{
    int k=1;
    int element=0;
    int seen=-1;
    printf("\n\nElements with Frequency Greater than %d: \n",k);

      for (int i = 0; i < l; i++)
      {
        int count = 0;
        for (int j = 0; j <l; j++) {
            if (arr[i] == arr[j])
                count++;

        }

        if (count>k)
        {
            element++;
        }
```

```
    }
    printf("%d",element);
}

int elements_with_frequency_lower(int arr[],int l)
{
    int k=2;
    int element=0;
    int seen=-1;
    printf("\n\nElements with Frequency Lower than %d: \n\n",k);

      for (int i = 0; i < l; i++)
      {
        int count = 0;
        for (int j = 0; j <l; j++) {
            if (arr[i] == arr[j])
                count++;

        }

        if (count<k)
        {
            element++;
        }
    }
    printf("%d",element);

}


int no_of_elements_greater_than_k(int arr[],int l)
{   int k=5;
    int count =0;
    for(int i=0;i<l;i++)
    {

        if(arr[i]>k)
        count++;
    }
    printf("\n\nNumber of elements greater than %d:\n%d\n",k,count);
```

```
    return 0;
}
int no_of_elements_lower_than_k(int arr[],int l)
{
    int k=2;
     int count =0;
    for(int i=0;i<l;i++)
    {

        if(arr[i]<k)
        count++;
    }
    printf("\nNumber of elements less than %d:\n%d\n",k,count);
    return 0;
}


float maximum_value(int arr[],int l)
{
    printf("\nMaximum value among the array elements:\n");
    descending_sort(arr,l);
    return arr[0];
}

float minimum_value(int arr[],int l)
{
    printf("\n\nMinimum value among the array elements:\n");
    ascending_sort(arr,l);
    return arr[0];

}

int no_of_positive_elements(int arr[],int l)
{
    int count=0;
    for(int i=0;i<l;i++)
    if(arr[i]>=0)
    count++;
    printf("\nNumber of Positive elements:\n %d",count);
}
```

```
int no_of_negative_elements(int arr[],int l)
{
    int count=0;
    for(int i=0;i<l;i++)
    if(arr[i]<0)
    count++;
    printf("\nNumber of Negative elements:\n %d",count);
}


int main()
{
    int arr[5]={3,3,3,3,1};
  int l = sizeof(arr) / sizeof(arr[0]);
  printf("Enter length of array:");
  scanf("%d",&l);
  printf("Input Array:\n\n");
  for(int i=0; i<l;i++)
  {
    printf("Enter %d element of array:\t",i);
    scanf("%d",&arr[i]);
  }


ascending_sort(arr,l);

descending_sort(arr,l);

no_of_even(arr,l);

no_of_odd(arr,l);

no_of_positive_elements(arr,l);

no_of_negative_elements(arr,l);



float addition=sum_of_elements(arr,l);
```

```
printf("%.2f\n\n",addition);

float average=mean_of_elements(arr,l,addition);
printf("%.2f",average);

frequency_of_each_element(arr,l);

elements_with_frequecncy_greater(arr,l);

elements_with_frequency_lower(arr,l);

maximum_frequency(arr,l);

minimum_frequency(arr,l);


no_of_elements_greater_than_k(arr,l);

no_of_elements_lower_than_k(arr,l);

int r1=maximum_value(arr,l);
printf("\nMaximum Value:\n %d",r1);

int r2=minimum_value(arr,l);
printf("\nMinimum Value:\n %d",r2);


return 0;
}
```

end

0.4 Output-

```
PS C:\Users\yugal\pp> ./miniproject
Enter length of array:5
Input Array:

Enter 0 element of array:       1
Enter 1 element of array:       1
Enter 2 element of array:       4
Enter 3 element of array:       2
Enter 4 element of array:       4
Asceending Sorting:
1
1
2
4
4

      Descending Sorting:
      4
      4
      2
      1
      1

      Number of even elements:
      3

      Number of odd elements:
      2


      Number of Positive elements:
       5
      Number of Negative elements:
       0
```

```
Sum of elements:
12.00


Mean of elements:
2.40

 Frequency of each element:
----------------------
 Element| Frequency
----------------------
4       |       2
2       |       1
1       |       2
----------------------


Elements with Frequency Greater than 1:
4

Elements with Frequency Lower than 2:

1

Number of elements greater than 5:
0

Number of elements less than 2:
2
```

```
Maximum value among the array elements:

Descending Sorting:
4
4
2
1
1

Maximum Value:
 4

Minimum value among the array elements:
Asceending Sorting:
1
1
2
4
4

Minimum Value:
 1
PS C:\Users\yugal\pp>
```

0.5 Profiling-

C: > Users > yugal > Downloads > profyugal

```
Flat profile:

Each sample counts as 0.01 seconds.
 no time accumulated

  %   cumulative   self              self     total
 time   seconds   seconds    calls  Ts/call  Ts/call  name
  0.00      0.00     0.00       18     0.00     0.00  swap
  0.00      0.00     0.00        1     0.00     0.00  ascending_sort
  0.00      0.00     0.00        1     0.00     0.00  descending_sort
  0.00      0.00     0.00        1     0.00     0.00  elements_with_frequecncy_greater
  0.00      0.00     0.00        1     0.00     0.00  elements_with_frequency_lower
  0.00      0.00     0.00        1     0.00     0.00  frequency_of_each_element
  0.00      0.00     0.00        1     0.00     0.00  maximum_value
  0.00      0.00     0.00        1     0.00     0.00  mean_of_elements
  0.00      0.00     0.00        1     0.00     0.00  minimum_value
  0.00      0.00     0.00        1     0.00     0.00  no_of_elements_greater_than_k
  0.00      0.00     0.00        1     0.00     0.00  no_of_elements_lower_than_k
  0.00      0.00     0.00        1     0.00     0.00  no_of_even
  0.00      0.00     0.00        1     0.00     0.00  no_of_negative_elements
  0.00      0.00     0.00        1     0.00     0.00  no_of_odd
  0.00      0.00     0.00        1     0.00     0.00  no_of_positive_elements
  0.00      0.00     0.00        1     0.00     0.00  sum_of_elements

 %         the percentage of the total running time of the
```

0.6 Miscellaneous Data-

Starting Date -14/11/22

Starting Day -Monday

Ending Date -16/11/22

Ending Day -Wednesday

Total Time required - 2 days

Total line of code - approx 350 lines of code

Total number of functions - 15 functions used

Language Used - C Language

Profiller used - Gprof

Debugger used - GDB

Project Title - Arr[ ] Array