# Assignment

K. Geetika
AP19110010123
CSE-G

1. Take the elements from the user and sort them in descending order and do the following

a) Using Binary search find the element and the location in the array where the element is asked from user.

b) Ask the user to enter any two locations print the sum and product of values at those locations in the sorted array.

Program:-

```
#include<stdio.h>
int main()
{
 int number[100];
 int c, first, last, middle, search, i, n, j, a;
 Printf("enter the value of N\n");
 scanf("%d",&n);
 Printf("enter the numbers\n");
 for(i=0;i<n;++i)
   scanf("%d",&number[i]);
 for(i=0;i<n;++i)
 {
  for(j=i+1;j<n;++j)
   {
     if(number[i]<number[j])
     {
       a = number[i];
       number[i] = number[j];
       number[j]=a;
     }
   }
 }
Print("The number arranged in descending order are given below\n");
```

```
for(i=0; i<n; ++i)
{
  printf("%d\n", number[i]);
 }
Printf("enter value to find\n");
scanf("%d", &search);
first=0;
last=n-1;
middle=(first+last)/2;
while (first<=last) {
 if (number[middle]< search)
  first=middle+1;
 else if (number[middle]==search){
  Printf("%d found at location %d.\n", search, middle+1);
  break;
 }
 else
   last=middle-1;

 middle = (first+last)/2;
}
if (first>last)
  Print("not found! %d isn't present in the list.\n", search).

  return 0;
}.
```

2. Sort the array using merge sort where elements are taken from the user and find the product of kth elements from first and last where k is taken from the user.

Program :-

```
#include <stdio.h>
Void merge sort(int[], int, int, int);
Void partition (int[], int, int);
int main()
{
  int list[50];
  int i, size, v, Pro=1;
  Printf ("enter total number of elements:");
  Scanf ("%d", &size);
  Printf("enter the elements: \n");
  for(i=0; i<size; i++)
  {
     scanf ("%d", &list[i]));
  }
  Partition (list, 0, size-1);
  Printf ("After merge sort: \n");
  for(i=0; i<size; i++)
  {
    Print f("%d ", list[i]);
  }
  Print f("enter the k value");
  Scanf ("%d", &v);
  for(i=0; i<=v, i++)
  { if(i<=v){
     Pro=Pro* list[i];
  }
  Printf ("\n\t%d", Pro);
}
  retun 0;
}
```

```
Void Partition (int list [], int , low, int high)
{
 int mid ;
 if(low< high)
 {
  mid=(low+high)/2 ;
  Partition (list , low, mid );
  Partition (list , mid+1, high);
   mergesort(list , low, mid, high);
Void mergesort (int list [], int low, int mid , int high )
{
 int i, mi, k, lo , temp [50],
  lo = low ;
  i = low ;
  mi= mid +1 ;
  while ((lo <= mid) && (mi <= high))
  {
   if (list [lo] <= list [mi])
   {
    temp[i ]= list [lo] ;
     lo++ ;
   }
   else
    {
     temp[i ]= list [mi] ;
     mi++ ;
    }
     i++ ;
  }
  if(lo>mid )
  {
    for (k=mi, k <= high, k++)
 if (lo > mid )
 {
   for (k= mi; k<= high; k++)
    {
```

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

3. Discuss insertion sort and selection sort with examples.

Program:-

a)
```
#include<stdio.h>
#include<conio.h>
#define size 5
void insertion_sort(int arr[], int n);
void main()
 {
    int arr[size],i, n;
    Printf("\n enter the number of elements in the array:");
    Scanf("%d", &n);
    Printf("\n enter the elements of the array:");
    for(i=0; i<n; i++)
    {
      Scanf("%d", &arr[i]);
    }
    insertion_sort(arr, n);
    Printf("\n the sorted is:\n");
    for(i=0; i<n; i++)
    Printf("%d\t", arr[i]);
    getch();
 }
void insertion_sort(int arr[], int n)
{
```

```
int i, j, temp;
for (i=1; i<n; i++)
{
    temp=arr[i];
    j = i-1;
    while((temp<arr[j]) && (j>=0))
    {
        arr[j+1]=arr[j];
        j--;
    }
    arr[j+1] = temp;
}
```

b)

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
int smallest(int arr[], int k, int n);
void selection_sort(int arr[], int n);
void main(int argc, char *argv[]){
    int arr[10], i, n;
    Printf("\nEnter the number of elements in the array:");
    Scanf("%d", &n);
    Printf("\nenter the elements of the array:");
    for(i=0; i<n; i++)
    {
        scanf("%d", &arr[i]);
    }
    Selection_sort(arr, n);
    Printf("\nthe sorted array is:\n");
    for(i=0; i<n; i++)
    Print("%d\t", arr[i]));
}
```

```
Void selection_sort (int arr[], int n)
{
    int k, pos, temp;
    for (k=0; k<n; k++)
    {
        pos=smallest (arr, k, n);
        temp=arr[k];
        arr[k] = arr[pos];
        arr[poss] = temp;
    }
}
```

4. Sort the array using bubble sort where elements are taken from the user and display the elements

i) in alternate order

ii) Sum of elements in odd positions and product of elements in even positions

iii) Elements which are divisible by m where m is taken from the user

Program:-

```
#include <stdio.h>
void main()
{
int a[100],n,i,j,temp, sum=0,prod=1, m;
Printf("enter number of elements\n");
Scanf("%d,&n);
Printf ["enter %d integers\n", n);
for (i=0;i<n;i++)
{
 Scanf("%d",&a[i]);
}
for (j=0; j<n-i-1;j++)
{
 if (a[j]>a[j+1])
```

```
{
  temp = a[j]
  a[j] = a[j+1];
  a[j+1] = temp;
  }
 }
}
Print("\n sorted list in ascending order:\n");
for(i=0;i<n;i++)
 {
  Printf("%d\n",a[i]);
 }
 Printf("the alternate order is");
 for(i=0;i<n;i++)
 {
  if(i%2==0)
   {
    printf("%d",a[i]);
   }
  }
   for(i=0;i<n;i++)
   {
     if(i%2!=0)
      {
       sumo=sumo+a[i];
      }
    }
    Printf("\n sum of odd index is %d", sumo);
    for(i=0;i<n;i++)
    {
     if(i%2==0)
```

```
{
  prod=prod*a[i];
  }
}
printf("\n product of odd index is %d", prod);
printf("\n enter the value of m\n");
scanf("%d", &m);
for(i=0; i<n; i++)
{
  if(a[i]%m==0)
  {
   printf("%d", a[i]);
  }
}
}
```

5. Write a recursive program to implement binary search?

Program:-

```
#include <stdio.h>
#include <stdlib.h>
int Binary search (int arr[], int num, int first, int last)
{
if(first>last)
printf("number you have entered is not found");
}
else
int mid;
mid = (first+last)/2;
if (arr[mid]==num)
{
```

```
else
{
printf("element you have asked for is found at index %d",mid);
exit(0);
}
else if (arr[mid]>num)
{
Binary search (arr, num, first, mid-1);
}
else
Binary search (arr, num, mid+1, last);
}
}
}
int main()
{
int arr[] = {100, 130, 150, 170, 110};
int num = 130;
int first = 0, last = (size of (arr) / size of (arr[0])) - 1;
Binary search (arr, num, first, last);
}
```

Output:-

element you have asked for is found at Index 2