DSA ASSIGNMENT-6
(Searching and Sorting)

chandanam. Tanmayee
AP19110010402
CSE-F

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.

```
#include<stdio.h>
int binarySearch (int arr[], int a, int b, int x)
{
    if(b>=a)
    {
        int mid=a+(b-a)/2;
        if(arr[mid]==x)
            return mid;
        if (arr[mid]>x)
            return binarySearch(arr,a,mid-1,x);
            return binarySearch(arr,mid+1,b,x);
    }
    return -1;
}
int main()
{
    int num;
    printf("Enter the size of array: ");
    scanf("%d", &num);
    int i,j,a,val[num],op,var,p1,p2,sum,pro;
```

```
for (a=0; a<num; a++)
{
    printf ("Enter value: ");
    scanf ("%d", &val[a]);
}
for(i=0; i<num; ++i)
{
    for (j=i+1; j<num; ++j)
    {
        if (val[i] < val[j])
        {
            a = val[i];
            value[i] = val[j];
            val[j] = a;
        }
    }
}
printf ("Array in descending order : ");
for (i=0; i<num; i++)
{
    printf ("%d", val[i]);
}
printf ("\n *** OPERATION_LIST ** \n");
printf ("1. Find value at entered position \n 2. Find the
        position of element \n 3. Printing sum of multiplication
        of values at entered positions,");
printf ("\n Enter choice : \n");
scanf ("%d", &op);
switch (op)
{
    case 1:
        printf ("Enter the position to obtain value:");
        scanf ("%d", &var);
```

```
        printf ("The value at %d position is %d", var, val[var]);
        break;
    case 2:
        printf ("Enter element to find position: ");
        Scanf ("%d", &var);
        int result = binarySearch (val, 0, num-1, var);
        (result == -1) ? printf ("Element is not present
        in array"); printf ("Element is present at index
        %d", result);
        return 0;
    case 3:
        printf ("\n Enter two positions to find sum and product
                of values \n");
        scanf ("%d %d", &p1, &p2);
        Sum = val [p1] + val[p2];
        pro = val[p1] * val [p2];
        printf ("SUM = %d \n", sum);
        printf ("MULTIPLICATION = %d", pro);
        break;
    }
}
```

OUTPUT:

```
Enter the size of the array: 4
Enter value : 24
Enter value : 56
Enter value : 25
Enter value : 75
Array in descending Order : 75562524
```

*** OPERATION_LIST ***

1. Find the value at Entered position
2. Find the position of element
3. Printing sum & multiplication of values at entered positions

Enter choice:

1

Enter the position to obtain value: 3

The value at 3 position is 24.

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.

```
#include<stdio.h>
#include <stdlib.h>
void merge (int arr[], int I, int m, int r)
{
    int i,j,k;
    int n1 = m-I+1;
    int n2 = r-m;
    int L[n1], R[n2];
    for (i=0; i<n1; i++)
    L[i] = arr[I+i];
    for (j=0; j<n2; j++)
    R[i] = a11 [m+1+j];
        i=0;
        j=0;
        k=I;
        while (i<n1 && j<n2)
        {
            if (L[i] <=R[j])
```

```
    {
      arr[k]=L[i];
       i++;
    }
    else
     {
       arr[k] = R[j];
        j++;
     }
     k++;
  }
  while (i<n1)
  {
      arr[k] = L[i];
        i++;
        k++;
   }
   while (j<n2)
   {
       arr[k] = R[j];j++;k++;
    }
}
void mergeSort (int arr[], int l, int r)
{
    if(l<r)
    {
       int m = l + (r-l)/2;
       merge(arr,l,m,r);
    }
}
void printArray (int A[], int size)
{
   int i;
   for(i=0;i<size;i++)
    print("%d",A[i]);
```

```
        printf("\n");
    }
int main()
{
    int size, v;
    printf("Enter array size: ");
    scanf("%d", &size);
    int val[size];
    for(v=0; v<size; v++)
    {
        printf("Enter value : ");
        scanf("%d", &val[v]);
    }
    printf("Given array is \n");
    printArray(val, size);
    mergeSort(val, 0, size-1);
    printf("\n sorted array is \n");
    printArray(val, size);
    int k, f, I, p1, p2, temp;
    printf("Enter the value of k to find the product
            of elements from first and last :");
    scanf("%d", &k);
    p1 = p2 = 1;
    for(f=0; f<=k; k++)
    {
        temp = val[f];
        p1 *= temp;
    }
    for(I = size-1; I >= k; I--)
    {
        temp = val[I];
        p2 *= temp;
    }
```

printf("product of kth elements from first and last one: %d %d", p1, p2);

Output:

Enter the array size: 4

Enter value: 1

Enter value: 2

Enter value: 3

Enter value: 4

Given array is

1 2 3 4

Sorted array is

1 2 3 4

Enter the value of k to find the product of elements from first and last: 4

Product of kth elements from first and last are: 4 8 12 16.

3) Discuss Insertion sort and selection sort with examples.

Insertion Sort:

One element from the array is selected & is compared to the one side of the array and inserted to the proper position while shifting the rest of the elements accordingly.

Example:

The lower part of an array is maintained to be sorted. An element which is to be inserted in this sorted sub-list, has to find its appropriate place & then it has to be inserted there, hence the name Insertion sort.

Selection sort:

Selection sort is a simple sorting algorithm. This sorting algorithm

is an in-place comparison based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty & the unsorted part is the entire list.

Example:

Consider the array:

[8,4,3,2]

The first element is 8. The next part we must find the smallest number from the remaining array. The smallest number from 4 3 and 2 is 2. So, we replace 8 by 2.

The new array is [2,4,3,8]. Again, this process is repeated.

finally, we get the sorted array as [2,3,4,8].

Program

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

(i) in alternative order

(ii) sum of elements in odd positions & product of elements in even positions.

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

```
#include<stdio.h>
void bubbleSort(int arr[], int n)
{
    int i, j, temp;
    for (i=0; i<n-1; i++)
    for (j=0; j<n-i-1; j++)
    if (j==0 ar[j] > ar[j+1])
    {
```

```
            temp = ar[j];
            ar[j] = ar[j+1];
            ar[j+1] = temp;
        }
    }
int main()
{
    int size, i;
    printf("Enter size of required array : ");
    scanf("%d", &size);
    int arr[size];
    for(i=0; i<size; i++)
    {
        printf("Enter element : ");
        scanf("%d", &arr[i]);
    }
    bubbleSort(arr, size);
    printf("sorted array : \n");
    for(i=0; i<size; i++)
    {
        printf("%d", arr[i]);
        printf("\t");
    }
    printf("\n\t**MENU**\n");
    printf("1.Display elements in alternate order \n");
    printf("2. Sum of elements in odd positions and Product
            of elements in even positions \n");
    printf("3. Divisible by m \n");
    int op, sum=0, product=1, m;
    printf("Enter choice: ");
    scanf("%d", &op);
    switch(op)
    {
```

```
case 1:
for(i=0; i<size; i+=2)
 {
    printf ("%d \t", arr[i]);
 }
 case 2:
  for(i=0; i<size;i+=2)
  {
     sum=sum +arr[i];
  }
  for(i=1 ;i<size; i+=2)
  {
     product =product *arr[i];
  }
  printf ("sum : %d \n",sum);
  printf ("Product : %d \n" ,product );
  case 3:
     printf("Enter value m : ");
     Scanf ("%d", &m);
    printf ("Numbers divisible by %d are : \n",m);
  for(i=0; i<size ;i++)
  {
     if (arr[i] % m==0)
     {
        printf ("%d \t", arr[i]);
     }
  }
 }
}
```

Output : Enter size of required array : 5
Enter element : 14
Enter element : 36
Enter element : 85
Enter element : 47
Enter element : 96
sorted array :
14 36 47 85 96

/** MENU **/

1. Display elements in alternative Order.
2. Sum of elements in odd positions and Product of elements in even positions
3. Divisible by m

Enter choice: 2

sum: 157

Product : 3060

Enter value m : 2

Numbers divisible by 2 are :

14 36 96

5. Write a recursive program to implement binary Search

```
#include <stdio.h>
#define MAX_LEN 10
int binary_Search_recursive (int l[], int arrayBegin, int arrayEnd, int a)
{
    int m, pos;
    if (arrayBegin <= arrayEnd)
    {
        m = (arrayBegin + arrayEnd) / 2;
        if (l[m] == a)
            return m;
            else if (a < l[m])
             return binary_search_recursive (l, arrayBegin, m-1, a);
          else
            return binary_search_recursive (l, m+1, arrayEnd, a);
      }
      return -1;
}
void read_list (int l[], int n)
{
    int i;
    printf ("\nEnter the elements : \n");
    for (i=0; i<n; i++)
        Scanf ("%d", &l[i]);
}
void print_list (int l[], int n)
{
    int i;
    for (i=0; i<n; i++)
        printf ("%d \t", l[i]);
}.
```

```
Void main()
{
    int l[MAX-LEN], num, ele, f, l1, a;
     int pos;
     printf ("In Binary Search Using Recursive method");
      printf ("In Enter the number of elements : ");
       scanf ("%d", &num);
        read_list (l, num);
       printf ("In Elements present In the list are : In In ");
        print_list (l, num);
       printf ("InIn Enter the element you want to search : InIn");
        scanf ("%d", &ele);
{
     printf ("In Recursive method: In");
       pos = binary_search_recursive (l, 0, num, ele);
       if (pos == -1)
       {
          printf ("Element is not found");
       }
        else
        {
          printf ("Element is found at %d position", pos);
        }
       }
}
```

Output

```
Binary Search Using Recursion method
Enter the number of elements : 3
Enter the elements
8 76
5 46
8 90
```

Elements present in the list are:

876 546 890

Enter the element you want to search:

546

Recursive method:

Element is found at 1 position.