ASSIGNMENT-6

P. Ruthik Venkat
AP19110010549

1) Take the Elements from the user and sort them in descending order and do the following:

(a) uping Binary search find the Element and the location in the array where the Element is asked

(b) Ask the user to Enter any two locations and print the sum and product of values at those location is Sorted array.

Explanation :

The aim of this problem is to Create an array and Soft the Elements of array in descending order after sorting using Binary Search find the Elements and location and then the users must Enter two locations and find the Sum and product of given two locations.

Algorithm :

→ start

→ Create an array and Enter the Elements

→ Give the Conditions in the program to Sort Elements

→ Give the Elements you want to Search

→ then give Condition Enter two locations and print the sum of those two and producer of those two.

→ stop

```
#Pnclude < stdio.h>
void sovt (int a[], int n)
{
  int i, j, temp;
  for (i=0; kn; i++)
  {
    for (j = i+1 ; j<n; j++)
    {
      if (a[i] < a[i])
      {
        temp = a[i]
        a[i] = a[j]
        a[j] = temp;
      }
    }
  }
}
int binary (int a[], int e, int n)
{
  int i=0; j=n-1; mid;
  while (i<=j)
  {
    mid (i+j)/2 ;
    if (a[mid] == e)
        return mid +1;
    else
    {
      if (e<a[mid])
          j = mid -1;
      Else i = mid +1;
    }
```

```
}
if (i>j)
{
  return 0;
}

}
int main()
{
int n, i, a[20], f, e, m1, m2;
printf("Enter the number of Elements of array");
scanf ("%d", &n);
printf ("Enter the Element in a array");
&Scanf("%d",
for (i=0; i<n ; i++)
     scanf ("%d", &a[i]);
sort (a,n);
for (i= 0; i<n; i++)
    scanf ("%d", &a[i]);
printf ("Enter the elements in the array to find ");
scanf ("%d", &e);
f = binary (a, e, n);
if (f != 0)
{
  printf ("Element is found at %d position", f);

}
else
{
  printf ("Element is found at %d position", f);
}
else
{
```

```
    printf("Element not found");
  }
  printf("Enter two positions in the array to find the sum and product");
  Scanf("%d %d", &m1, &m2);
  m1---;
  m2---;
  printf("the sum is %d", a[m1] + a[m2];
  printf("the product is %d", a[m1]*a[m2];
}
```

Sample output :-

Enter the no. of Elements of array - 5

Enter the Elements of array - 5
6
4
7
8
2

Elements in descending order - 8 7 6 5 4 2

Enter the Element to find in array - 3

Element not found

Enter the position of array to find sum and product - 1
2

the sum 14 and the product is 48.

Explanation of output :-

the output of this program is printed by satisfying all the conditions and then the sum and product is printed by choosing the positions in the array.

2) Sort the array using merge sort where Elements are taken form the user find the product of kth Elements from 1st and last where k is taken from the user

Explanation of Problem:

the aim of this program is to sort the Elements in the array either in ascending (or) descending order using merge sort and than after sorting the user will given position and by giving position the product of first and the fill position will be printed.

Algoritham:

→ start
→ Create an array and Enter the Elements
→ using merage sort, sort them into ascending order
→ give the Condition to give the position
→ And then print the product from 1st to last (until position)
→ stop.

```
# include < stdio.h >
void merage (int * a, int s, int e)
{
  int temp[15];
  int i, j, k, mid;
  mid = (e+s)/2;
  i = s
  j = mid + 1;
  k = s;
  while (i <= mid && j <= e)
  {
    if (a[i] < a [j])
    {
```

```
        temp[k++] = a[i++]
    }
    else
    {
      temp [k++] = a [i++];
    }
  }
  while (j <= e)
  {
    temp[k++] = a [j++]
  }
  while (i <= mid)
  {
    temp [k++] = a [i++];
  }
  for (i=s; i<e ; i++)
  {
    a [i] = temp [i];
  }
}
void merage (int * a, int s, int e)
{
  int m;
  if (s<e)
  {
    m=(s+e)/2
    mergesort (a, s, m)
    mergesort (a, m+1, e);
    merge (a, s, e)
  }
}
int main()
{
  int n, i, a[20], Prod1=1, Prod2=1, k, j, c=1;
  printf("Enter the No. of Elements in array");
```

```
    scanf("%d", &n);
    printf("Enter the Elements of array");
    for(i=0; i<n; i++)
        scanf("%d", &a[i]);
    merge sort (a, 0, n-1)
    for(i=0; i<n; i++)
    {
      printf("%d", a[i]);
    }
    printf("Enter position till which you want to find product");
    scanf("%d", &k);
    i=0
    d=n-1;
    while(c<=k)
    {
      prod1 = prod1 * a[i];
      prod2 = prod2 * a[i];
      i++;
      d--;
      c++;
    }
    printf("%d %d", prod1, prod2);
}
```

Sample Input and output:

Enter the no. of elements of array - 6

Enter the element of array - 6
9
5
4
8
2

After merging - 2,4,5,6,8,9

Enter the position till which you want to find product - 3

product is 40 432

Explantion of output:

The output of this program is printed by satisfying all the condition given in the program and prints the product from first and last

---

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 position and product in Even position

(iii) Elements which are divisible by m and where m is given by the user

Explanation of Problem:

the aim of this problem is to print the elements using bubble sort and then printing them in alternate order and printing sum and product and the number which are divisible

Algorithm:

→ start

→ Enter the size and elements in the array

→ write the conditions for using Bubble sort

→ and the print the Element By all the orders

→ stop.

```
#include <stdio.h>
void main()
{
 int a[100], n, i, j, temp, sum=0, prod=1, m;
 printf("Enter Number of Elements\n", n)
 Scanf("%d", &n);
 printf("Enter %d integers \n", n)
 for(i=0; i<n; i++)
 {
  Scanf("%d", &a[i]);
 }
 for(i=0; i<n-1; 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] = tem;
   }
  }
 }
 printf("Sorted list in ascending order");
 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)
  {
   Sum= Sum+a[i];
  }
}
printf ("Sum of odd index is %d", Sum);
for (i=0; i<n; i++)
{
  if (i%2 == 0)
  {
    prod = prod*a[i];
  }
}
pvintf ("product of even index is %d", prod);
printf ("Enter the value of m");
sCanf ("%d", &m);
for (i=0; i<n; i++)
  {
    if (a[i]%m == 0)
      {
        printf ("%d", a[i]);
      }
  }
}
```

Sample input and output:

Enter the no. of elements - 6

Enter 6 integers = 6
5
4
3
2
7

Sorted list in ascending order is - 2
3
4
5
6
7

The alternate order is 2
4
6

Sum of odd index is 8

product of Even index is 48

Enter the value of m - 3

3, 6

Explantion of output:

the output of this program is printed by Satisfy all the Conditions and checks all the laps and prints in all the Conditions given below.

5) write a recursive program to implement binary search

Explanation of Problem:

the aim of this program is to implement binary search by using recursive function that means by calling many times. the value this program is done

Algorithm:

→ start
→ Enter the Elements in the array
→ write the Conditions for binary using recursive function
→ Call the function
→ print the sorted Elements
→ Stop.

```
#include <stdio.h>
void sort (int a[], int n)
{
  int i, j, temp;
  for (i=0; i<n; i++)
  {
    for (j=i+1; j<n; j++)
    {
      if (a[i] > a[j])
      {
        temp = a[i]
        a[i] = a[j]
        a[j] = temp;
      }
    }
  }
}
```

```
int binary (int a[], int e, int i, int j)
{
  int mid = (i+j)/2;
  if (a[mid] = e)
        return mid+1;
  if (i>j)
        return 0;
  else
  {
     if (e < a (mid))
          return binary (a,e, i,mid-1)
     else
          return binary (a,e,mid+1, i);
  }
}
int main ( )
{
  int n,i,a[20], f, e;
  printf ("Enter the no. of Elements of array");
  Scanf ("%d", &n);
  printf ("Enter the Elements of array\n");
  for (i=0; i<n; i++)
        Scanf ("%d", &a[i]);
  Sort (a, n);
  for (i=0; i<n; i++)
        printf ("%d", a[i]);
  printf ("Enter the Element to find in array");
  Scanf ("%d", &e)
  f = binary (a, e, 0, n-1);
  if (f!=0)
  {printf (Element is found at position %d", f);
```

```
else
{
printf ("Element not found");
}
}
```

Sample input and output:

Enter the no. of elements of array - 5

Enter the Elements of array - 5

8

3

4

1

8

Enter the Element to find in array 1

the Element is found at 5 position

Explanation of output:

the output of this program is printed by satisfying all the Conditions and By Calling the function (recursive) many times the output is printed

3) Discuss Insertion sort and selection sort with Examples

Insertion sort:

* Insertion sort is simple sorting algorithm that works the way we sort playing cards in our hands

* Algorithm

// sort an arr[] of size n

insertion sort (arr, n)

loop from i=1 to n-1

---- a) Pick element arr[i] and insert it into sort sequences arr[0--i-1]

Example:

12, 11, 13, 5, 6

Let us loop for i=1 (second element of the array) to 4 (last element of the array)

i=1. since 11 is smaller than 12, move 12 and insert 11 before 12

11, 12, 13, 5, 6.

i= 2. 13 will remain at its position as all elements in A[0---1-1] are smaller than 13

11, 12, 13, 5, 6.

i = 3. 5 will ~~remain~~ move to the begining and all other elements from 11 to 13 will move one position ahead of their current position

5, 11, 12, 13, 6

i=4. 6 will move to position after 5 and elements from 11 to 13 will moove one position ahead their current position

5, 6, 11, 12, 13

Selection sort:

The selection sort algorithm sorts an array by repeatedly finding the min elements (considering ascending order) from unsorted part and putting it at the beginning. the algorithm maintains two subarrays in a given array.

1) the subarray which is already sorted.
2) remaining subarray which is unsorted.

In every iteration of selection sort, the min element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray.

Example:

arr [] = 64 25 12 22 11

// find the min element in arr [0....4]
// and place it at begining

11 25 12 22 64.

// find the min element in arr [1---4]
// and place it at begining arr [1---4]

11 12 22 25 64

// find the min element in arr [2---4]
// and place it at begining arr [2---4]

11 12 22 25 64

// find the min element in arr [3---4]
// and place it at begining arr [3---4]

11 12 22 25 64