Assignment-6

DSA

D.SATYAMANI SYAM
AP19110010631
CSE-P

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

a. Using binary search find the elements and location in the array what the element is asked from user.

b. Ask the user to enter any two location print the sum and products of values at those location in sorted array

```
#include <stdio.h>
int binary search (int a[], int m, int n, int y).
{
 if(n>=m){
  int mid = m+(n-a)/2;
  if (a[mid]==y).
    return mid;
  if(a[mid]>y)
    return binary search(a, m, mid-1, y);
  return binary search(a, mid+1, n, y);
  }
  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);
    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]
     val[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 the value at entered position\n2. Find the
position of element\n3. Printing Sum & 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 the 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;
```

2 Sort the array using Merge Sort whose elements are taken from the user & find the product of k^{th} elements from 1st & last where k is taken from the user

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

```
    }
    else
    {
      arr[k] = R[j];
      j++;
    }
    k++;
  }

  while(i<n1)
  {
    a[k] = L[i];
    i++;
    k++;
  }
  while(j<n2)
  {
    arr[k] = R[j];
    j++;
    k++;
  }
}
void mergeSort(int a[], int l, int r).
{
  if(l<r)
  {
    int m = l+(r-l)/2;
```

```
    mergeSort(a, l, m);
    mergeSort(a, m+1, r);

    merge(a, l, m, r);
    }
}

void printArray(int A[], int size)
{
    int i;
    for(i=0; i<size; i++)
        printf("%d", A[i]);
    printf("\n");
}

int main()
{
    int siz, v;
    printf("Enter array size: ");
    scanf("%d", &siz);
    int val[siz];
    for(v=0; v<siz; v++)
    {
        printf("Enter value: ");
        scanf("%d", &val[v]);
    }
    printf("Given array is \n");
    printArray(val, siz);
```

```
mergeSort(val, 0, siz-1);
printf("\n Sorted Array is \n");
printArray(val, siz);
int k, f, l, 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; f++)
{
 temp = val[f];
 p1 *= temp;
}
for(i=siz-1; l>=k; l--)
{
 temp = val[l];
 p2 *= temp;
}
printf("product of k^th element from first & last are: %d %d", p1, p2);
}
```

3. Discuss Insertion Sort and selection Sort with example.

Ans: Insertion Sort:

It is a simple sorting algorithm that builds the final sorted list by transferring one element at a time. Transfers an element at a time to the partially sorted array, more efficient than selection sort Complex than selection sort.

example:

13, 12, 14, 6, 7

Let us loop for i = 1 to 4.

i = 1. Since 12 is smaller than 13, move 13 and insert 12 before 13

12, 13, 14, 6, 7

i = 2. 14 will remain at its position as all elements in A[0 1] are smaller than 14.

12, 13, 14, 6, 7.

i = 3; 6 will move to the begining and all other elements from 12 to 14. will move one position ahead of their current position

6, 12, 13, 14, 7

i = 4. 7 will move to position after 6, & elements from 12 to 14 will move one position ahead of their current sitheation.

6, 7, 12, 13, 14

Selection Sort

A simple sorting Algorithm that repeatedly searches remaining items to find the smallest elements & moves it to the correct location. Finds the least element & moving it accordingly

less efficient than insertion sort - Simple than insertion sort

Examples:

arr[]: 65 26 13 23 12

* find the minimum element in arr[0...4]
* and place it at beginning

12 26 13 23 65

* Find the min element in arr [1...4]
* and place it at beginning

12 13 26 23 65

* Find the min element in arr[2...4]
* and place it at begining at 11

12 13 23 26 65

* Find the min element in arr[3...4]
* and place it at beginning at 11

12 13 23 26 65

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

i. in alternate order

ii. Sum of elements in odd positions & product of elements in every positions

iii. elements which are divisible by m where m is taken from the user

Ans:

```
#include<stdio.h>
/*Bubble sort function*/
void bubblesort(int ar[], int n).
{
  int i,j, temp;
 for(i=0; i<n-1; i++)
 for(j=0; j<n-i-1; j++)
 if(ar[j] > ar[j+1]).
 {
   temp = a[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 ar[size];
```

```
for(i=0; i<size; i++)
{ printf("Enter element: ");
scanf("%d", &arr[i]);
}
bubble_sort(arr, size);
printf("Sorted array: \n");
for(i=0; i<size; i++)
{
  printf("%d ", arr[i]);
  printf("\t");
}
printf("\n/** MENU **/\n");
printf("1. Display elements in alternate order.\n");
printf("2. Sum of elements in odd positions and product of elements
      in even position\n");
printf("3. Divisible by m\n");
int op, sum=0, product=1, m;
printf("Enter choice: ");
scanf("%d", &op);
{
  case 1;
  for(i=0; i<siz; i+=2)
  { printf("%d\t", arr[i]);
  }
  case 2;
  for(i=0; i<siz; 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]);
      }
    }
  }
}
```

5. Write a recursive program to implement binary search.

```
#include <stdio.h>
int binary search(int a[], int l, int h, int x)
{
  int mid = (l+h)/2;
  if(l>h) return -1;
  if(a[mid]==x)
  return mid;
  if(a[mid]<x)
  return binary search(a, mid+1, h, x);
  else
  return binary search(a, l, mid-1, x);
}
int main(void)
{
  int a[100];
  int siz, pos, val;
  printf("Enter length of the array");
  scanf("%d", &siz);
  printf("In Enter array elements:\n");
  for(int i=0; i<siz; i++)
  scanf("%d", &a[i]);
  printf("Enter element to search\n");
  scanf("%d", &val);
  pos = binary search(a, 0, siz-1, val);
  if(pos<0)
    printf("Cannot find the
    element in array\n", val);
  else
  printf("The position of %d in the
  array is: %d\n", val, pos+1);
  return 0;
}
```