Assignment - 6

B. Sridhar
CSE - F
AP19110010447.

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

a) using BS to find element and location in array, where the element is asked.

b) ASK the user to enter the 2 locations, print the sum & product of values at locations in sorted array.

```
# include <stdio.h>
Void sort (int arr[], int n){
    int i, j, temp;
   for(i=0; i<n; i++){
    for(j=i+1; j<n; j++){
     if (arr[i] < arr(j)){
        temp = arr(i);
         ar[i] = arr(j);
         arr[j] = temp;
   }
   }
   }
   }
int binary (int arr[], int n, int x) {
    int i=0, j=n-1, middle;
   while (i<=j) {
    middle = (i+j)/2;
  if (arr[middle] == x) {
      return middle+1; }
```

```
else {
  if (x< arr[middle]) j= middle-1;
 else  i=middle+1;
}
 if (i>j) {
 return 0; }
}
int main () {
 int z, x, arr[50], a, b, c, d;
 Printf("Enter the No.of elements to array");
 Scanf("%d", &z);
 Printf(" Enter the elements to array\n");
 Scanf("%d" &
 for(x=0; x<n; x++) {
  Scanf("%d", & arr[x]);
  Sort (arr, z);
 Printf("%d is descending order", sort (arr, z));
  for(x=0 ; x<z; x++){
  Printf("%d", arr[x]);
  Printf(" Enter the element to findinthe array\n");
  Scanf("%d",&b);
  a= binary(arr,b,z);
  if(a!=0) {
  Printf("The Element is found at %d Position", a); }
  else {
  Printf(" The element is missing\n"); }
  Printf(" Enter the position of array to find the
          Sum and Product : \n");
  Scanf("%d\n %d", &c, &d);
```

```
    c--; d--;
    Printf(" The sum here is %d", arr[c] + arr[d]);
    Printf(" The Product here is %d\n", arr[c] * arr[d]);
}
```

O|T:

```
Enter the no.of elements to array : 4
Enter the elements to array:
3
4
5
1
5431 is in descending order.
Enter the element to find in the array: 4
The Element is found at 1 position
Enter the position of array to find the Sum and Product:
2
3
The Sum here is 6
The product here is 5
```

2) Sort the array using Merge sort where elements are taken from the user and find the product of the k^{th} elements from first & last taken from user.

```
#include <stdio.h>
#include <stdlib.h>
    void merge (int arr[], int a, int b, int c) {
      int x, y, z;
```

```
int p= b-a+1;
int q= c-b;
int A[p], B[q];
for(x=0; x<p; x++){
   A[x] = arr[a+x];
for (y=0; y<q; y++){
   B[y] = arr [b+1+y];
   x=0;
   y=0;
   z=1;
while (x<p && y<q){
if( A[x] <= B[y]){
   arr[z] = A[x];
   x++; }
else{
arr[z] = B[y];
   y++; }
   z++; }
while ( y< q){
arr[z] = B[y];
   y++;
   z++;
}
}
Void mergesort (int arr[], int a, int c){
   if(a<c){
   int b=a+(c-1)/2;
   merge sort (arr, a, b);
   merge sort ( arr, b+1, c);
   merge sort (arr, a, b, c); }
}
```

```
Void deployArray (int arr[], int space) {
    int x;
    for(x=0; x< space; x++)
    Printf("%d", arr[x]);
    Printf("\n");
}
int main() {
    int space, n, m;
    Printf(" Enter the Size of the array: ");
    Scanf("%d",&n);
    int val[n];
    for(m=0; m<n; m++) {
    Printf(" Enter the value to array:\n");
    Scanf("%d", & val[m]);
    }
    Printf(" The entered value is : ");
    Scanf("%d", & val[m]);
    }
    Printf(" The Entered given array is : \n");
    deployArray (val, space);
    mergeSort( val, 0, n-1);
    Printf(" The Sorted Array is \n");
    deployArray (val, n);
    int K, a, b, temp, x, y;
    Printf(" Enter the K value: ");
    Scanf("%d", & K);
      x=y=1;
```

```
for(a=0; a<=k; a++){
 temp = val[a];
  x*=temp;
}
 for(b=n-1; b>=k; b--){
 temp = val[b];
  y*=temp;
 }
 Printf(" The product of kth elements from the array are:
                                %d %d", x,y);
}
```

O/T:

```
Enter the Size of the array: 4
Enter the value to array: 6
 Enter the value to array: 5
 Enter the value to array: 4
 Enter the value to array: 9
The Entered array is
  6 5 4 9
The Sorted array is
  4 5 6 9
Enter the k Value: 20
The product of kth elements from the array are: 0 1
```

3) Insertion Sort: This works by inserting the set of values in the existing sorted file. It can construct the sorted array by inserting a single element at a time. This process continuous until a whole array is sorted with in the time. in the same order. It works on a simple sorting algorithm which works in the way of we sort of playing cards in our hands. The insertion sort method save an effective amount of memory.

Advantages:

i) It is faster than other sorting algorithms.

ii) It is easily implemented, fast, very efficient on the sets of data.

Example:

a) Pick element arr[i] and insert it into sorted sequence of arr[0 ... i-1]

Ex: 2, 8, 1, 0,

for i = 1 to 4 (last element in array)

Since, i=1 8 is larger than 1, move 8 and insert 1 after 8.

This is, 2, 1, 8, 0.

for i = 2 to 4

Since, i=2, 8 is larger than 0, move 8 and insert 0 after 8.

This is, 2, 1, 0, 8.

Since, i=3 , 0 is smaller than 1, so it is sorted to

This is,

2, 0, 1, 8.

Since, i= 1 , 0 is smaller than 2, so it is sorted to

This, 0, 2, 1, 8.

Since, i= 3, 1 is smaller than 2, So it sorted to

This, 0, 1, 2, 8.

Here done by insertion sorted list 0, 1, 2, 8.

b) Selection Sort: This works on the array by the repeatedly minimum element from unsorted part to putting at beginning. The main process of searching is minm key plaing until all the elements are placed at right position.

This algorithm maintains 2 subarrays for a given array:

i) The subarray which is already sorted.

ii) The remaining subarray is unsorted.

Example: →(0 1 2 3 4)

8 3 9 2 19

1→ 8 3 9 2 19

2→ 2 3 9 8 19

3 → 2 3 8 9 19

4) #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 ( arr[j] > arr [j+1]) {
    temp = arr[j];
    arr[j] = arr[j+1];
    arr[j+1] = temp;
  }
 }
Void main() {
  int a, b;
  Printf(" Enter the size to array:");
  Scanf("%d", &a);
  int arr[a];
  Printf(" Enter the elements to array:\n");
  for(b=0; b<a; b++){
    scanf("%d", &arr(b]); }
  bubblesort (arr, a);
  Printf ("The Sorted Array is : \n");
  for( b=0; b<a; b++) {
    Printf ("%d", arr(b]);
    Printf ("\t"); }
  Printf("\nMain MENU\n");
  Printf (" I. Show the elements in alternate order\n");
  Printf (" II. Sum of elements in odd Positions and Product
          of elements in even positions\n");
  Printf (" III. Divisible by 2\n");
  Printf (" IV. Exit\n);
```

```
int choice, Sum=0, Product = 1, z ;
while(1) {
    Printf(" Enter your choice: ");
    Scanf("%d", &choice);
    Switch(choice) {
  case 1:
    for(b=0; b<a; b+=2) {
     Printf("%d\t", arr[b]); }
  case 2:
     for(b=0; b<a; b+=2) {
     ~~Pri~~ Sum = Sum+ arr[b]; }
      for(b=1; b<a; b+=2) {
       Product = Product *arr[b]; }
      Printf ("The Sum is : %d\n", Sum);
      Printf (" The Product is : %d\n", Product);
  case 3:
      Printf(" Enter the Value for z : ");
      Scanf("%d", &z);
      Printf("The Numbers are divisible by %d are : \n", z);
       for(b=0; b<a; b++){
       if(arr[b] % z == 0) {
        Printf("%d\t", arr[b]); }
  case 4}:
     } Printf(" Exit");
  Default: Printf(" Invalid Entry");
        }
     }
```

O/T:

```
Enter the size to array: 3
Enter the elements to array:
3
4
1
The Sorted Array is
1  3  4
Main MENU
i. Show the elements in alternate ~~mode~~ order:
ii. Sum of elements in odd positions and products of elements
        in even positions
iii. Divisible by 2
iv. Exit
Enter your choice: 1
1  4
Enter your choice: 4
```

5) Write a recursive program to implement binary search.

```
#include <stdio.h>
int Binarysearch (int arr[], int a, int b, int c){
    int middle = a+b/2;
    if (a>b) {
     return -1; }
    if (arr[middle] == c) {
      return middle; }
    if (arr[middle] < c) {
     return Binarysearch (arr, middle+1, b, c); }
```

```
    else
    return BinarySearch (arr, a, middle-1, c);
}
int main() {
    int arr[50];
    int i, x, y, z;
    Printf ("Enter The Size of the array: ");
    Scanf ("%d", &x);
    Printf (" Enter the elements to the array: \n");
    for(i=0; i<x; i++) {
    Scanf ("%d", &arr[i]);
    Printf (" Enter the element for to search: ");
    Scanf ("%d", &z);
    y = BinarySearch (arr, 0, x-1, z);
    if (y<0)
    Printf (" The Position of element %d is can't find, Try Again!");
    else
    Printf (" The Position of %d in array is %d \n", z, y+1);
    return 0;
    }
```

O/T:

```
Enter the Size of the array: 3
Enter the elements to the array:
9
5
2
Enter the element for to search:
5
The Position of 5 in array is 2.
```