Assignment-6

Kavuru Manasa

AP19110010343

CSE-H

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

(a) Using Binary search find the elements and the location in array where the elements 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>
void binary search();
int number [20];
void main ()
{
        int number [20];
        int i, j, a, n;
        Printf ("enter numbers")
        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;
              }
          }
     }
   Printf (" Number in desending order");
   for (i=0 ; i<n; i++)
   {
      Print f (" %d\n" number[i]);
   }
   Printf (" enter two locations");
     int x, y, sum, Product;
     scanf (" %d,%d", &x, &y);
     sum = number [x] + number[y]
    Product = number [x]* number [y]
 Printf(" sumof numbers in two locations is %d", sum);
```

```
        Printf (" Product of two numbers is %d", Product);
        binary search();
}
void binary search()
  {
        int c, first , last, middle search;
        Printf (" enter the value to search");
        scanf ("%d", & search);
        first 0;
        last n-1;
        middle = (first + last)/2
        while (first < = last) {
            if (number [Middle] < search) {
                    first = middle +1;
              }
        elce if (number [middle]== search)
          {
            printf("%d found at %d", search, middle+1)
          }
        else
          {
              last = middle-1
              middle = (first+last)/2;
          }
        if (first > last)
            printf ("%d is not inlist", search);

}
```

OUTPUT:-

```
Enter numbers 5

8
6
9
5
4
Numbers in desending order
9
8
6
5
4
Enter two locations 2 3

sum of numbers in two locations is 14

Product of two numbers is 48

Enter the value to search 8

8 found at 2
```

② sort the array using Merge sort where elements are taken from the user and find the Product of k^{th} elements from first and last where K is taken from the user

```
# include < stlib.h>
# include < stdio.h>
void merge (int arr[] , int l, intm, intr)
{
    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] = arr [l+i]
    for (j=0; j<n2 ; j++)
        R[j] = arr [m+1+j]
    i=0 ;
    j=0;
    k= l;
    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++;
        }
}
void merge sort (int arr[], int l, int r)
{
    if (l< r)
    {
        int m= l + (r-l)/2;
        merge sort (arr, l, m);
        mergesort ( arr, m+1, r);
        merge (arr, l, m, r);
    }
}
```

```
void Print-Array (int A[], int size)
{
    int i;
    for(i=0; i<size; i++){
        Printf ("%d", A[i]);
    Print("\n")
    }
}
int main()
{
    int arr[5];
    int i;
    int arr_size = size of (arr) / size of (arr[0]);
    for(i=0; i< arr_size; i++) {
        Printf (" enter the elements");
        scanf ("%d", & arr [i]);

 }
 Printf (" Given array is\n");
 Printf(~~Array~~ (arr, arr_size));
 merge sort (arr, 0, arr_size -1);
 Printf ("\n sorted array is\n");
 Printf (" Enter the values of k");
 scanf ("%d", & k);
 int from first = arr[k-1];
int fromlast = arr[5-k];
 printf ("%d", fromlast * fromfirst);
 return 0;
}
```

OUTPUT:-

```
Enter the elements 3
Enter the elements 6
Enter the elements 4
Enter the elements 2
Enter the elements 9
Given array is
3 6 4 2 9
sorted array is
9 6 4 3 2
Enter the value of k2
18
```

③ Discuss Insertion sort and selection sort with examples.

Insertion sort:-

Insertion sort works by ~~intersecting~~ inserting the set of values in the existing sorted List. It constructs the sorted array by ~~intersecting~~ Inserting a single element at a time: This Process continues untill while array is sorted in some order.

The Primary concept behind insertion sort is to insert each item into its appropriate place in the final list. The insertion sort method saves an effective amount of memory.

<u>Working of insertion sort</u>

-> It uses two sets arrays where one stores the sorted data and other on unsorted data.

-> The sorting algorithm works untill there are elements in the unsorted sets.

After each interation it chooses the first element of the sorted partition and in serts it into the proper place in the sorted set.

Advantages of insertion sort

→ Easily implemented and very effecient when used with small sets or data. The addional memory space requried of insertion sort is les (ie O(1)); It is considered to be live sorting technique as the list can be sorted as the new elements are received.

Example

25	15	30	9	99	20	26

15	25	30	9	99	20	26

15	25	30	9	99	20	26

9	15	25	30	99	20	26

9	15	25	30	99	20	26

9	15	20	25	30	99	26

9	15	20	25	26	30	99

Definition of selection sort:-

The selection sort perform sorting by searching for the minimum value number and placing it into the first or last position according to the order. The Process of searching minimum key and placing it in the Proper Position is continued untill the all elements are placed at right Position.

Working of the selection sort

→ suppose an array ARR with N elements in the memory

→ In the ~~second~~ first Pass, again the smallest key is searched along with its Position then the ARR[Pos] is swapped with Arr[0]. Therefore ARR[0] is sorted

→ In the second Pass, again the ~~sm~~ Position of the smallest value is determined in the subarray of N-1 elements interchange the ARR[Pos] with ARR[1]

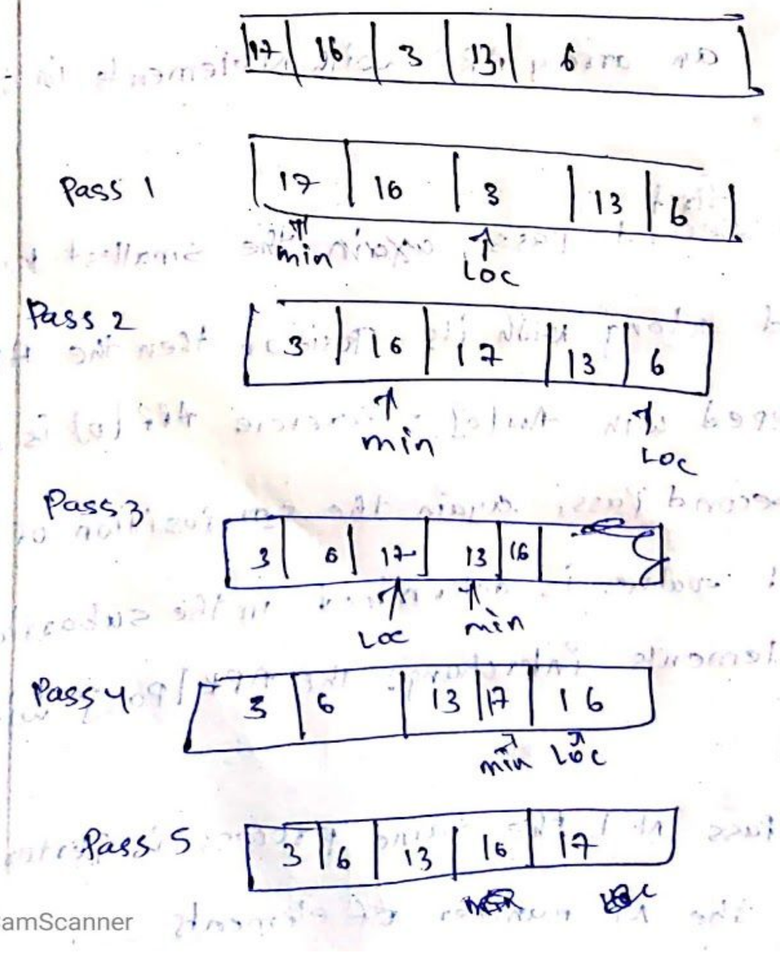
→ In the Pass N-1 the same process is Performed to sort the N number of elements

Advantages of selection sort

The main advan-tages of selection sort is that it Performs well on a small list.

Further , because it is an in-place sorting algorithm no additional temporary storage is requried beyond what is need to hold the original list . .

example



(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 of elements in even Position

iii, Elements which are divisble by m where m i's taken from user.

```
# include < stdio.h>
void main()
{
  int a[100], n, i, j, temp, sum= 0, Produ= 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 (i=0; i<n-1; i++)
  {
      for (j=0; j<n-1; j++)
```

```
            {
             if (a[i] > a[i+1])
             {

               temp= a[i]
                 a[i] = a[i+1]
                 a[i +1] = temp;
              }
          }
      }
      printf ("\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 even inder is %d", Prodi;
Printf ("\n Enter the value m\n");
scanf ("%d", &m);
for(i=0; i<n; i++)
{
    if (a[i] % m ==0)
    {
        Printf ("%d", a[i]);
```

```
            }
        }
    }
```

OUTPUT

```
Enter number of elements 5
Enter 5 integers
1 9 4 7 8
sorted list in ascending order:
1 0
4 1
7 2
8 3
9 4
The alternate order is 1 7 9
Sum of odd index is 12
Product of ~~odd~~ even index is 63
Enter value of m 2
4 8
```

(5) write a recursive program to implement search.

```
# include < stdio.h>
int recursive Binary search (int arr[], int _start_index;
                        int end _index, int element)·
  {
     if (end_index >= start _index)
      {
        int (middle)= start_ index + (end-index - start_index)/2
        if (array [middle] = = element)
          return middle;
        if (array [middle]> element)
        return recursive binarysearch (array, start_index,
                             middle -1, element)
        return recursive binarysearch (array , start,
                             middle+1, end index, element
      }
     return -1;
  }
int main (void) {
       int array[ ] = {1, 4, 7, 9, 16, 56, 70};
       int n = 7;
int element = 9;

int found_index = recursive binarysearch (array,
                                    0, n-1, search
                                          element);
if (found_index == -1) {

    Printf ("Element not found in the array");

}

else{.
   Printf (" Element found at index : %d",
                              found_index);
}

return0;

}
```

OUTPUT

```
Element found at (location) index : 3
```