Assignment-6

AP19110010380
CSE-H

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 main()
{
 int i, low, high, mid, n, key, arr[100], temp, i, one,
                                two, sum, product;

 printf("Enter the number of elements in array");
 scanf("%d", &n);
 printf("Enter %d integers", n);
 for(i=0; i<n; i++)
 scanf("%d", &arr[i]);
 for(i=0; i<n; i++)
 {
  for(j=i+1; j<n; j++)
  {
```

```
        if(arr[i]<arr[j])
        {
          temp=arr[i];
          arr[i]=arr[j];
          arr[j]= temp;
        }
    }
}
printf("Enter value to find");
 scanf ("%d", &key);
  low = 0;
  high=n-1;
  mid = (low+high)/2;
  while(low<=high)
  {
    if (arr[mid]>key)
    low=mid+1;
    else if (arr[mid]==key){
     printf("%d found at location %d", key, mid+1);
     break;
  }
```

```
else
high=mid-1;
mid=(low+high)/2;
}
if (low>high)
{
 printf("Not found! %d isn't present in list.n",key);
}
printf("\n");
printf("Enter two locations to find sum & product");
scanf("%d",&one);
 Scanf("%d", &two);
 sum=(arr[one]+arr[two]);
 product= (arr[one]*arr[two]);
 printf("the sum of elements =%d ",sum);
 printf("the product of elements =%d",product);
 return 0;
}
```

Output :-

Enter number of elements in array 5
Enter 5 integers 9
7
5
4
3
Element of array is sorted in descending order:
9 7 5 4 3 .
Enter value to find 5
5 found at location 3
Enter two locations to find sum and product
of elements 3 5
The sum of elements = 8
The product of elements = 15 .

2. Sort the array using Merge sort where elements are taken from the user and find the product of values of Kth elements from first and last where K is taken from the user.

```
#include<stdio.h>
#include<conio.h>
#define MAX_SIZE 5
void merge_sort(int,int)
void merge_array(int,int,int,int);
int arr_sort[MAX_SIZE];
int main()
{
 int i, k, pro=1;
 printf("Simple Merge sort Example function & Array\n");
printf("\n Enter %d Elements for sorting \n", MAX_SIZE);
 for(i=0; i<MAX-SIZE; i++)
 scanf("%d", &arr_sort[i]);
 printf("\n Your Data :");
 for(i=0; i<MAX_SIZE; i++)
 {
   printf("\t %d", arr_sort[i]);
 }
```

```
merge_sort (0, MAX_SIZE - 1);
printf ("\n\n Sorted Data : ");
for (i=0; i<MAX_SIZE; i++){
   printf ("\t %d", arr_sort[i]);
}
printf (" Find product of Kth elements from first & last
        where k \n");
scanf ("%d", &k);
pro = arr_sort[k] * arr_sort [MAX_SIZE-k-1];
printf (" product = %d", pro);
getch();
}
void merge_sort (int i, int j) {
   int m;
if (i<j){
   m = (i+j)/2;
   merge_sort (i,m);
   merge_sort (m+1,j);
   // Merging two arrays
   merge_array (i,m,m+1,j);
}
}
```

```
void merge_array(int a, int b, int c, int d)
{
  int t[50];
  int i=a, j=c, k=0;
  while (i<=b && j<=d) {
    if (arr_sort[i] < arr_sort[j])
      t[k++] = arr_sort [i++];
    else
      t[k++] = arr_sort [j++];
  }
  // collect Remaining Elements.
  while (i<=b)
    t[k++] = arr_sort [i++];
  while (j<=d)
    t[k++] = arr_sort [j++];
  for(i=a, j=0; i<=d; i++, j++)
    arr_sort[i] = t[j];
  }
```

Output :-

Simple Merge sort Example - Functions & Array

Enter 5 elements for sorting

9
7
4
6
2

Your Data: 9 7 6 4 2

Sorted Data : 2 4 6 7 9

Find the product of kth elements from fast and last where k 2

Product = 36.

3) Discuss Insertion sort & Selection sort with examples.

Definition of Insertion sort :- Insertion sort works by inserting the set of values in the existing sorted file. It constructs the sorted array by inserting a single element at a time. This process continues untill the whole array is sorted in same order. The primary concept behind insertion sort is to insert each item into its appropriate place in final list. The insertion sort method saves an effective amount of memory.

Working of Insertion sort :-

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

→ The sorting algorithm works until there are elements in unsorted set.

→ Lets assume there are 'n' number elements in the array. Intially, the element with index 0 exists in the sorted set. Remaining elements in unsorted position of the set.

→ The first element of the unsorted position has array index 1.

→ After each iteration, it chooses the first element of the unsorted partition & inserts it into proper place in sorted set.

Advantages of Insertion sort :-

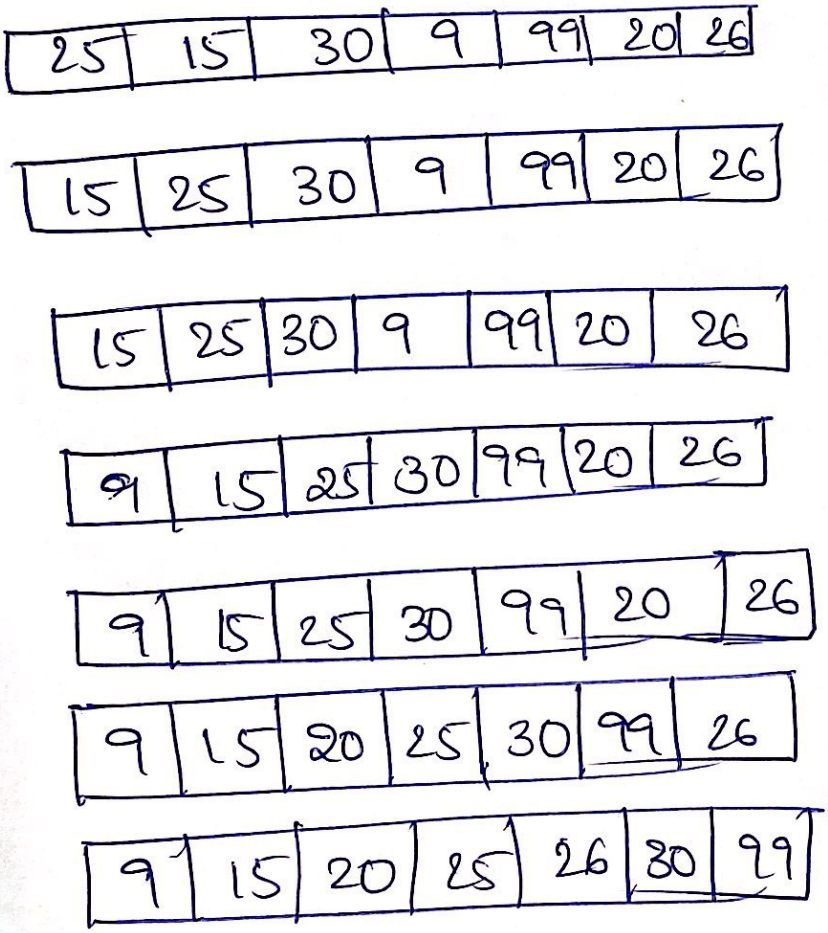
→ Easily implemented & very efficient when used with small sets of data.

→ The additional memory of space requirement of insertion sort is less (i.e; O(1)).

→ It is considered to be live sorting technique as the list can be sorted as the new elements are received.

→ It is faster than other sorting algorithms.

Example :-



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 order The process of searching the minimum key and placing it in the proper position is continued untill all the elements are placed at right position.

Working of Selection sort :-

→ Suppose an array ARR with N elements in the memory.

→ In the first pass, 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 position of the smallest value is determined in the subarray of N-1 elements interchange the ARR[pos] with ARR[1].

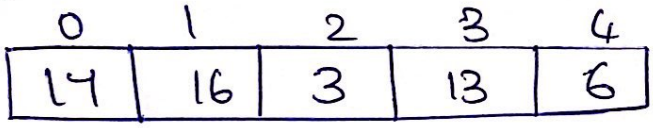
→ In the pass N-1, the same process is performed to sort the N number of elements.

Advantages of Selection Sort :-

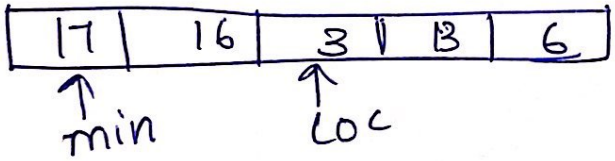
→ The main advantage of selection sort is that if performs well on a small list.

→ Further more, because it is an in-plane sorting algorithm, no additional temporary storage is required beyond what is needed to hold the orginal list.

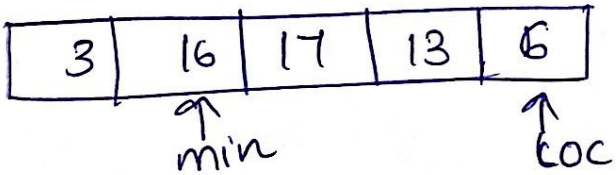
Example :-



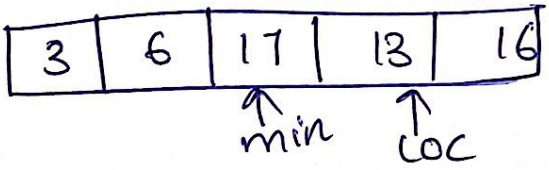
Pass 1



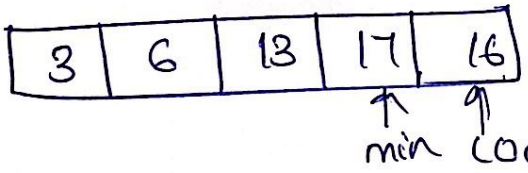
Pass 2



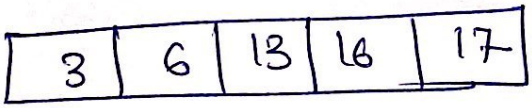
Pass 3



Pass 4



Pass 5



Complexity of Insertion sort :-

The best case complexeity of insertion sort is O(n) times, i.e when the array is previously sorted. In some way, when array is sorted in reverse order, the first element of unsorted array is to compared with each element in sorted sort. So, in the worst case, running time of insertion sort is quadratic i.e, $O(n^2)$. In average case also it has to make the minimum $(k-1)/2$ comparisions. Hence, the average case also has quadratic running time $O(n^2)$.

Complexity of Selection Sort :-

As the working of selection sort, does not depend on the original order of the elements in the array, so there is not much difference Between best case & worst case of selection sort.

The selection sort selects the minimum value element, in the selection process. All the 'n' number of elements are scanned; therefore n-1 comparisions are made in the first pass. Then the elements are interchanged. Similarly in the second pass also to find

the second smallest element we require scanning of rest n-1 elements & the process is continued till the whole array sorted.

Thus, running time complexity of selection sort is $O(n^2) = (n-1)+(n-2)+\dots+2+1 = n(n-1)/2 = O(n^2)$.

4. Sort the array using bubble sort where elements are taken from the user & display and find the elements
i, in alternate order
ii, Sum of elements in odd positions & product of elements in even position.
iii, Elements which are divisible by m where m is taken from the user.

```
#include<stdio.h>
#include<conio.h>
int main()
{
 int arr[50], i, j, n, temp, sum=0, product=1
 printf("Enter total number of elements to store:");
 scanf("%d", &n);
 printf("Enter %d elements:", n);
 for (i=0; i<n; i++)
    scanf("%d", &arr[i]);
 printf("\n Sorting array using bubble sort\n");
 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;
}
}
}
printf("All array elements sorted successfully \n");
printf("Array elements in ascending order:\n\n");
for(i=0;i<n;i++){
   printf("%d \n", arr[i]);
}
printf("Array elements in altenate order \n");
for(i=0;i<=n;i=i+2){
   printf("%d \n",arr[i]);
}
 for(i=1;i<=n;i=i+2){
      sum=sum+arr[i];
 }
```

```
printf("The sum of odd position elements are = %d \n",sum);
for(i=0; i<=n; i=i+2)
{
  product*=ar[i];
}
printf("The Product of even position elements are
                                = %d \n", product);
getch();
return();
}
```

Output :-

```
Enter total number of elements to store=5
Enter 5 elements
8
6
4
3
2
Sorting array using bubble sort
All array elements sorted successfully.
```

Array elements in ascending order:

2
3
4
6
8

Array element in alternate order:

2
4
8

The sum of odd position elements is 9

The product of even position elements are

6, 4.

5. Write a recursive program to implement binary search?

```
#include<stdio.h>
#include<stdlib.h>
void binarysearch(int arr[], int num, int first,
                  int last) {
int mid;
if (first >last) {
    printf("Number is not found");
}
else {
/* Calculate mid element */
mid = (first + last)/2;
 if (arr[mid]==num){
    printf(" Element is found at index %d", mid);
    exit(0);
}
else if (arr[mid] > num) {
    Binarysearch(arr, num, first, mid-1);
}.
else
{
```

```
    Binarysearch(arr, num, mid+1, last);
   }
  }
}
void main()
{
 int arr[100], beg, mid, end, i, n, num;
 printf("Enter size of array:");
  scanf("%d", &n);
 printf("Enter the values in sorted sequence \n");
 for(i=0; i<n; i++)
 {
  scanf("%d", &arr[i]);
 }
 beg=0;
 end=n-1;
 Printf("Enter a value to be search:");
  scanf("%d", &num);
  binary search(arr, num, beg, end);
}
```

Output :-

Enter the size of array :5

Enter values in sorted sequence

4
5
6
7
8

Enter a value to be search 5

Element is found at index 1.