Assignment - 6

1) Take the elements from the user and short them in decreasing 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) The user to enter any two locations print the sum and product of values at these locations in the sorted array.

Sol:

```
#include <stdio.h>
int main()
{
int, low, high, mid, n, key, arr [100], temp, i, one, two, sum,
product
    printf ("Enter the number of elements in array");
    scanf ("%d", &n);
    prints ("enter %d integers", n);
     for (i=0; i<n; i++)
     {
       if (i=0; i<n; i++)
      {
      if (j=i+1; j<n; j++)
      {
      if (arr[i] < arr [j])
      {
       temp = arr (i);
       arr (i) = arr (j);
       arr (j) = temp;
      }
    }
  }
}
```

```
printf("In Element of array is sorted in desending ording");
 for (i=0; i<n; i++)
{
 printf ("%d", arr [i]);
}
 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 the list, n", key);
   }
   printf ("\n");
   printf ('enter two locations to find Sum and product
                of the elements");
    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
2
Element of array is sorted in decreasing order;
9 7 5 4 2 Enter value to find 5
5 found at location 3
Enter two locations to find sum and product of the
elements 2
4
The sum of elements = 7
The product of elements = 10
```

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

Ans:-
```
# 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 functions and 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 ("In your Data:");
    For (i=0; i<MAX_SIZE; i++) {
        printf ("\t%d", area_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 the product of k^th elements from
            first and 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+1] = 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 and 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 elements from first and last where k

2

product = 36.

3) Discuss Insertion sort and selection sort with examples

Ans: 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 until 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.

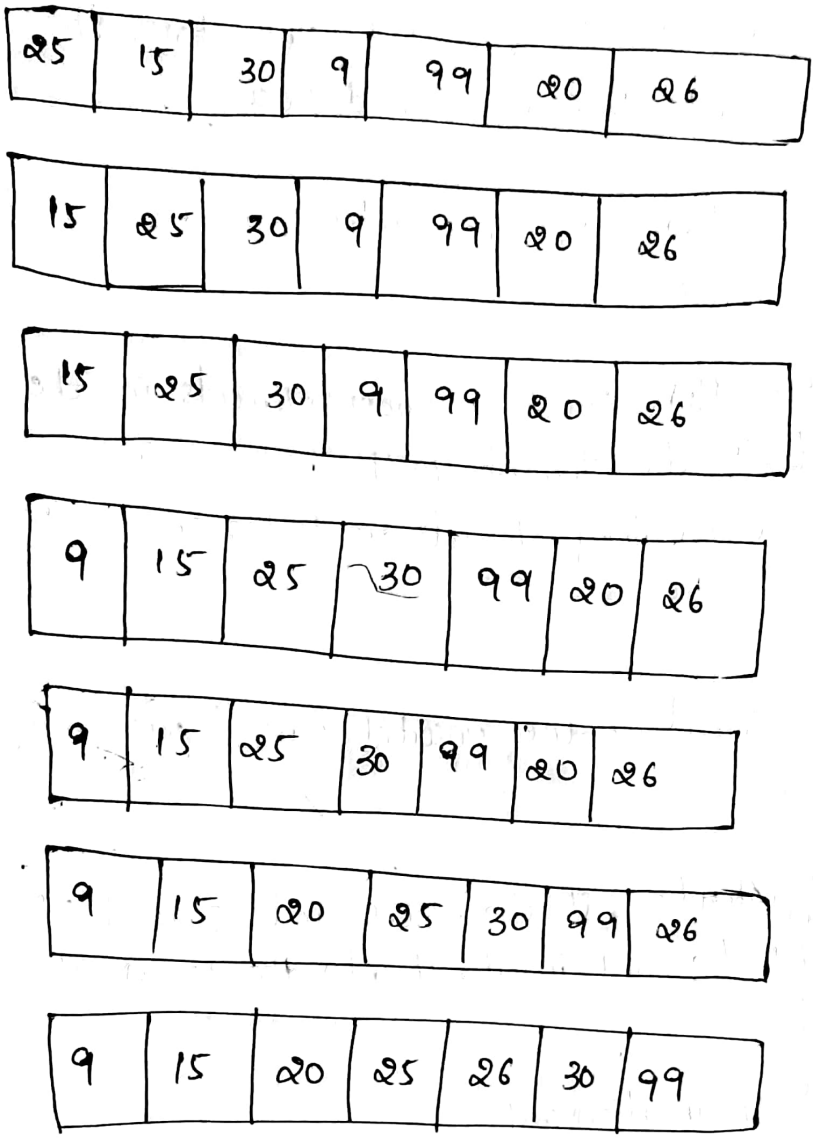
Working of insertion sort

* 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
* let's assume there are 'n' number elements in the array. intially, the element with index 0 (LB=0) exists in the sorted set Remaining elements are in the unsorted partition of the list.
* The first element of the unsorted partion has array index 1 (IF LB=0).
* After each interation, it chooses the first element of the insorted partition and inserts 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 additional memory space reauired of insertion sort is less (i.e; O(1)).
* It is considerd 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 the order (ascending or descending). The process of searching minimum key and placing it in the proper position is continued until the all elements are placed at right position.

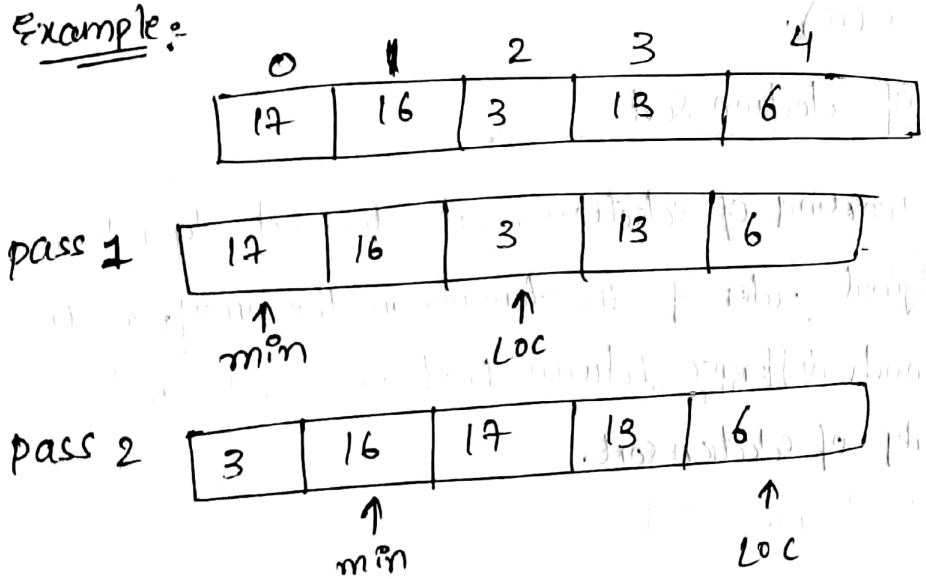
Working of the selection sort

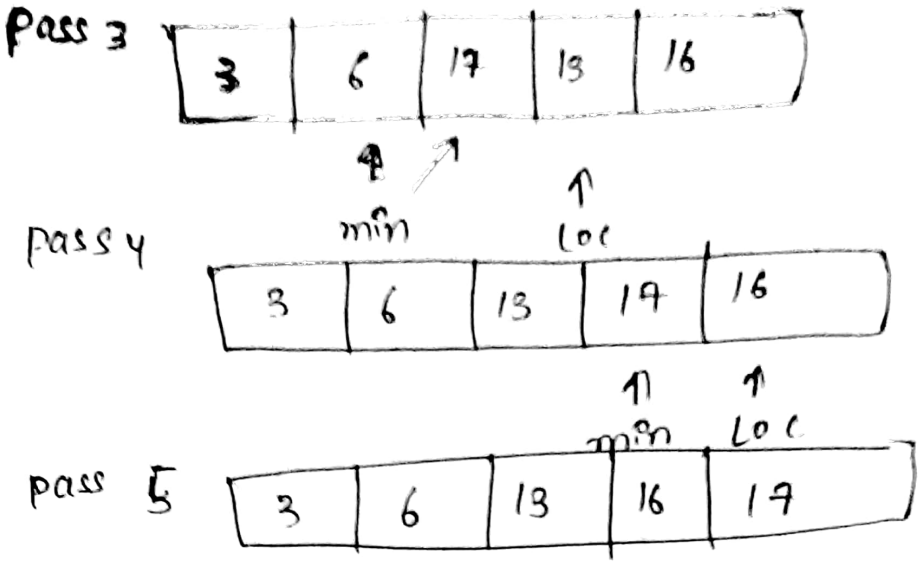
* Suppose an array ARR with N elements in the memory
* In the first pass, the smallest key is searched along with its position then the ARR(POS) is swapped with ARR(0). therefore, ARR(0) is sorted.
* In the second pass, again the position of the smallest value is determined in the subarray of N-1 elements 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 advantages of selection sort is that it performs well on a small list.
* Furthermore, because it is an in-place sorting algorithm, no additional temporary storage is reauried beyond what is needed to hold the original list.

Example:-





Complexity of insertion sort

The best case complexity of insertion sort is $O(n)$ times i.e. when the array is previously sorted. In the same way, when the array is sorted in reverse order, the first element of the unsorted array is to compared with each element in the sorted set 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 $(k-1)/2$ comparisons 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 and worst case complexity of selection sort.

The Selection sort selects the minimum value element in the selection process all the 'n' numbers of elements are scanned; therefore n-1 comparisons are made in the First pass. then the elements are interchanged. Similarly in the Second pass also find the Second Smallest element we Require scanning of rest n-1 elements and the process is continued till the whole array sorted.

Thus, running time complexity of selection sort is $O(n^2)$

$= (n-1) + (n-2) + \cdots + 2+1 = n(n-1)/2 = O(n^2)$.

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 positions and product of elements in even position.

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

Ans

```
#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 technique\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");
        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 *= arr[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 technique.
All array elements sorted sucessfully!
Array elements in ascending order.
2
3
4
6
8
Array elements in alternate order.
2
4
8
The sum of odd position element are=9
The product of even elements are=64
```

5) Write a recursive program to implement binary search?

Ans)

```
#include <stdio.h>
#include< stdlib.h>

void binary search (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 mid is eavual to number we are searching
        if (arr (mid) == num) {
          Binary search (arr, num, first, mid);
        & printf ("Elements is found at index %d", mid))
          Exit (0);
        }
        Else if (arr (mid] > num) {
          Binary Search (arr , num, first, mid-1);
        } Else{
            Binary Search (arr, num, mid+1, last);
      }
   }
}.
```

```
Void main ()
    int arr (100), beg, mid, end, i, n, num;
    printf ("Enter the size of an 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 the values in sorted sequence
4
5
6
7
8
Enter a value to be search : 5
Element is found at index 1
```