1)
```
# include <stdlib.h>
#include <stdio.h>

int comparator (const void* P1, const void* p2)
{
    return (* (int*) p2 -* (int*) P1);
}
int binary search (int arr [], int size, int search) {
        int beg = 0, end = site-1, mid;
        While (beg <= end) {
            mid = (beg + end) /2;
            if (arr [mid] == search) {
                return mid;
            }
            else if (arr[mid] < search) {
                end = mid-1;
            }
            else beg = mid + 1;
        }
        return -1;
}
int main ()
{
        int arr [100], site, search, i, Pos = -1, loc1, loc2;
        Printf ("\n Enter the site of the array (max 100)");
        scanf ("%d", & site);
        printf (" \n Enter the ~~s~~elements in array \n");
```

```
for (i=0; i< site; i++){
        scanf ("%d", &arr[i]);
  }
qsort (arr, site , siteof (int), comparator);
printf (" \n The sorted array is : \n");
    for (i=0; i< site;i++)
  {
    Printf ("%d", arr[i]);
    }
    printf ("\nEnter search element");
    Scanf (" %d", & search);
    POS = binary search (arr, site, search);
    if (POS = =-1) Print f (" Not found \n");
    else printf ("\n the %d search element is found at
      index %d \n", search, POS);
    Print f (" Enter two indexes\n");
    Scanf ("%d %d", & loc1, & loc2);
    Printf ("sum is %d \n", arr[loc1] + arr [loc2]);
    Printf ("Product is %d \n", arr [loc1]* arr[loc2]);
```
output :-

```
Enter the Site of the array (max 100) 5
Enter elements in array
5 2 3 6 7
The sorted array is :
7 6 5 3 2
Enter Search element 2
the 2 search element is found at index 4
```

```
Enter two indexes

2 3

sum is  8

product is 15
```

2)
```
# include <stdio.h>
# define ms 100
int a[ms];
void merge (int l1, int u1, int l2, int u2)
{
  int i,j, k, temp[ms];
   k=0;
   i=l1;
   j=l2;
   while ((i<=u1) && (j<=u2)) {
      if (a[i]<a[j]){
         temp[k]=a[i]; i++; k++;
      }
      else {
         temp[k] =a[j]; j++; k++;
         }
    }
    while (i<=u1){
        temp[k] =a[j]; j++; k++;
    }
    for (i=l1, k=0; i<u2; i++, k++){
         a[i] =temp[k];
    }
 }
 void mergesort (int lb, int ub){
    if (lb<ub)
    {
     int  mid =(ub+lb) /2;
```

```
        mergesort (lb, mid);
        merge sort (mid+1, ub);
        merge (lb, mid, mid+1, ub);
      }
  }
  int main () {
     int i, n, product = 1, k;
     printf ("\n Enter the size of the array max (100)");
     Scanf ("%d", &n);
     for (i=0; i<n; i++){
         Printf ("a[%d]\t =", i);
         Scanf ("%d", &a[i]);
     }
     printf ("\n The Product till the kth element is %d\n",
                                                   Product);
     return 0;
  }
```

Output:-

```
Enter the Size of the array 5
 a[0] = 1
 a[1]  = 6
 a[2]   = 1
 a[3]   = 54
 a[4]  = 2
Enter k
3
The product till the kth element is 2
```

3) Defination 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 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 ~~element~~ 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. Initially, the element with index 0 (CB=0) Exists in the sorted set remaining. Elements are in the unsorted Position has array index of the list.

* The first element of the ~~uns~~ unsorted Position has array index 1 (IF LB=0)

* After each interation, it chooses the first element of the insorted position & inserts it into the proper place in the sorted set.

Example :-

25	15	30	9	99	20	26

15	25	30	9	99	20	26

9	15	25	30	99	20	26

9	15	20	25	30	99	26

9	15	20	25	26	30	99

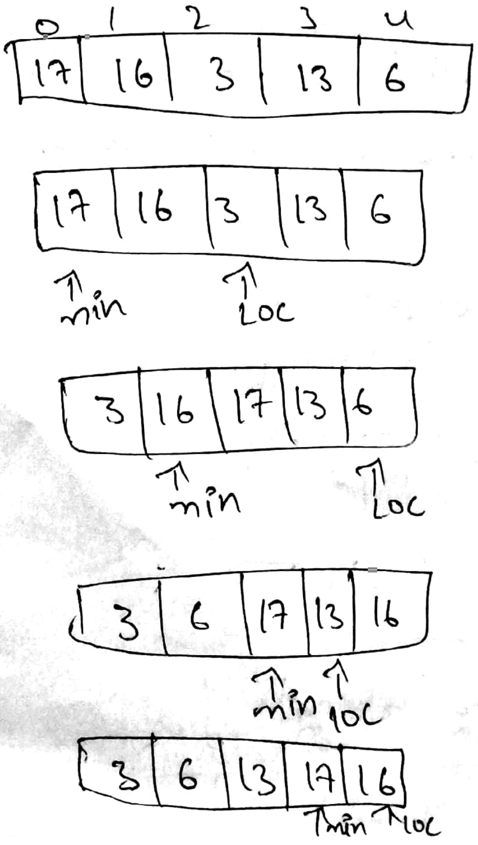
Definition of Selection Sort :

* The sort perform sorting by searching for the minimum value number & placing it into the first or last Position according to the order (ascending or descending). The Process of searching minimum key & 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 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, gain 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.

Example :-



3	6	13	16	17

4)
```
# include <stdio.h>
void displayAltsumPro(int arr[ ], int size){
    int i, sum =0, Product =1;
    printf ("Alternate elements \n");
    for (i=0; i<size ; i++) {
        if (i%2!=0){
            Product += arr[i];
        }
        else {
            Sum += arr[i];
            printf ("%d", arr[i]);
        }
    }
    printf ("\n sum of the odd elements = %d \n", sum);
    printf ("\n Product of the even elements= %d\n", Product);
}
void divM(int arr[], int size){
    int i=0, m;
    Printf ("Enter the m \n");
    scanf ("%d", &m);
    Printf ("elements divisible by %d \n", m);
    for (i=0; i<size ;i++){
        if (arr[i] %m ==0)
            Printf ("%d", arr[i]);
    }
}
void bubble sort (int arr[], int size)
{
    int i, j, temp;
    for (i=0; i<size-1; i++)
        for (j=0; j<size-i-1; j++)
```

```
            if (arr[j] > arr[j+1]) {
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        displayAltSumPro(arr, size);
        divM(arr, size);
}
int main ()
{
    int arr[100], size, i;
    Printf ("\n Enter the size of the array(max 100)");
    Scanf ("%d", &size);
    Printf ("\n Enter elements in array \n");
    for (i=0; i<size; i++) {
        Scanf ("%d", &arr[i]);
    }
    bubblesort (arr, size-1);
    return 0;
}
```

output :-

```
Enter the size of the array (max 100) 5
Enter elements in array
9 4 5 2 8
Alternate elements
2 5
Sum of the odd elements = 7
Product of the even elements = 14
```

```
Enter the m
3
elements divisible by 3
9
```

5)
```
# include <stdio.h>
int binary search (int arr[], int beg, int end, int search){
        int mid;
        if (beg <= end){
                mid= (beg+end) /2;
                if(arr[mid]==search) return mid;
                if (arr [mid] > search){
                        return binary search (arr, beg, mid-1, search
                }
                return binarySearch (arr, mid+1, end, search);
        }
        return -1;
}
int main ()
{
        int arr[100], size, search, i, Pos;
        Printf ("\n Enter the size of the array (max 100)"
        Scanf ("%d", &size);
        Printf ("\n Enter sorted elements in array \n");
        for (i=0; i<size; i++){
                scanf ("%d", &arr[i]);
        }
        Printf ("\n Enter search element ");
```

```
    Scanf ("%d", & Search);
    Pos = binarysearch (arr, 0, size-1, Search);
    if (Pos ==-1) Printf ("Not found \n");
    else Printf ("\n the %d Search element is found
                 at index %d \n", search, Pos);
    return 0;
}
```

Output :-

```
Enter the size of the array (max 100) 5
Enter sorted elements in array
1
2
3
4
5
Enter Search element 3
the 3 element is found at index 2
```