1:) 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 Point 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, j, 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("elements of array is sorted in descending order");
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 the 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 location to find sum and product
        of the element");
  Scanf ("%d", &one);
  Scanf ("%d", &two);
  Sum = (arr[one] + arr[two]);
  Product = (arr[one] * arr[two]);
  Printf ("the Sum=%d and the product=%d", Sum, Product);
   return 0;
}
```

2) Sort the array using merge sort where elements are taken from the user & find the product of kth elements from first & last where k is from user.

```
#include<stdio.h>
#include<conio.h>
#define MAX 5

void merge_sort(int, int)
void merge_array(int, int, int, int);
int arr_sort[MAX];
int main(){
    int i, k, pro=1;
Printf("Sample merge sort example function of array");
Printf("\n Enter %d elements for sorting\n", MAX);
for(i=0; i<MAX; i++)
Scanf("%d", &arr_sort[i]);
Printf("\n Your Data\n");
for(i=0; i<MAX; i++){
   Printf("%d", arr_sort[i]);
}
  merge_sort(0, MAX-1);
Printf("sorted data");
for(i=0; i<MAX; i++){
   Printf("%d", arr_sort[i]);
}
```

```
printf("Find the product of kth element from first
        and last when k\n");
scanf("%d", &k);
pro = arr_sort[k] * arr_sort[MAX - 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);
   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++];
    }
    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];
}
```

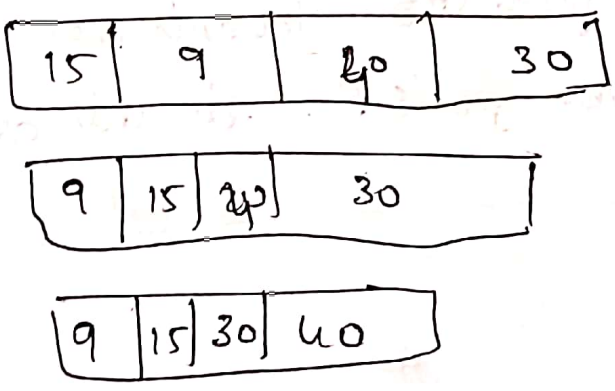
5) Discuss insertion sort and selection sort with Examples.

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 whole array is sorted in some order. The first concept of insertion sort is to insert each value into it's place in the final list. It saves an effective amount of memory.

advantages of insertion sort;

* Easily implemented and very efficient when used with small set of data's.
* It is faster than other sorting algorithms.

Example:



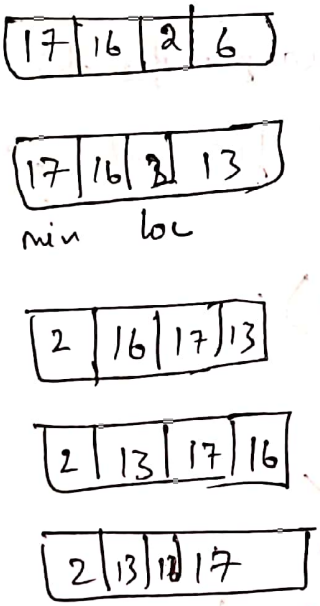
Complexity of inserted sort: The best case Complexity is $O(n)$ times. In the worst case. running time of insertion sort is $O(n^2)$ time ~~Worst case~~ Average case has $O(n^2)$

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 the minimum key and placing it in the proper position is continued until the all elements are placed at right positions.

advantages of selection sort

* The main advantage of selection is that it performs well on a small list.

Example:



Complexity of selection sort: The best case complexity of $O(n)$ times. Worst case complexity $O(n^2)$.

(10) Sort the array using bubble sort where elements are taken from user and display the elements in

i) in alternate order.

ii) Sum of elements in odd position and product of elements in even position.

iii) elements which is divisible by m where m is taken from 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("In sorting array using bubble sort technique");
for(i=0; i<(n-1); i++)
 {
   for(j=0; j<(n-2); j++)
   {
     if(arr[j] > arr[j+1])
     {
       temp = arr[j];
       arr[j] = arr[j+1];
       arr[j+1] = temp;
```

```
}
}
}
Printf("All array elements sted success fully\n");
Printf("array elements in ascending order:\n");
for(i=0; i<n; i++){
    Printf("%od", arr[i]);
}
Printf("array elements in alternat order\n");
for(i=0, i<=n; i=i+2){
    Printf("%od\n", arr[i]);
}
for(i=1; i<=n; i=i+2){
    Sum+=arr[i];
}
Printf("the sum of the odd position elements are %od", Sum
for(i=0; i<=n; i=i+2)
{
    Product*=arr[i];
}
Printf("the product of even position elements are
                                    =%od", Product);
getch();
return();
```

output:

enter total number of element to star: 4
enter 4 elements: 5

6
4
8

Sorting array using bubble sort technique.
all array elements sorted successfully:
array elements in ascending order:

4
5
6
8

array elements in alternate order

4
6

The sum of the odd position elements are = 13
The product of even position elements are = 24

(5) 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{
mid = (first + last)/2;
// if mid is equal to number we are searching
if(arr[mid] == num){
    Printf("Elements 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 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);
    Binarysearch(arr, num, beg, end);
  }
```

output:

Enter the size of an array 4

Enter the values in sorted sequence

5

6

7

8

Enter a value to be search: 7

elent is found at index 2