

```

1) #include <stdio.h>
void sort(int a[], int n)
{
    int i, j, temp;
    for (i = 0; i < n; i++)
    {
        for (j = 0 i + 1; j < n; j++)
        {
            if (a[i] < a[j])
            {
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
        }
    }
}

int binary(int a[], int e, int n)
{
    int i = 0, j = n - 1, mid;
    while (i <= j)
    {
        mid = (i + j) / 2;
        if (a[mid] == e)
            return mid + 1;
        else
        {
            if (e < a[mid])
                j = mid - 1;
            else
                i = mid + 1;
        }
    }
    if (i > j)
    {

```

```

    return 0;
}
}
int main()
{
    int n, i, a[20], f, e, m1, m2;
    printf("Enter no. of elements in array\n");
    scanf("%d", &n);
    printf("Enter the elements of array\n");
    for(i=0; i<n; i++)
    {
        scanf("%d", &a[i]);
    }
    sort(a, n);
    for(i=0; i<n; i++)
        printf("%d", a[i]);
    printf("Enter the element to find in array");
    scanf("%d", &e);
    f = binary(a, e, n);
    if (f != 0)
    {
        printf("element is found at %d position", f);
    }
    else
    {
        printf("element not found\n");
    }
    printf("Enter the position of array to find sum and product\n");
    scanf("%d %d", &m1, &m2);
    m1--;
    m2--;
    printf("the sum is %d", a[m1] + a[m2]);
    printf("the product is %d", a[m1] * a[m2]);
}

```


②

③

Selection Sort:- The selection sort algorithm sorts an array by repeatedly finding the minimum element from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in given array.

- 1) The subarray which is already sorted.
- 2) Remaining subarray which is unsorted.

In every iteration of selection sort, the minimum element from the unsorted subarray is picked and moved to the subarray.

Following example explains the above steps:

arr[] = 64 25 12 22 11

// Finding minimum element in arr[0, 4]

// and place it at beginning

// 25 12 22 64 11

// Finding minimum element in arr[1, 4]

// and place it at beginning of arr[1, 4] → // 11 12 25 22 64

// Finding minimum element in arr[2, 4]

// and place it at beginning of arr[2, 4]

// 11 12 22 25 64

// Finding minimum element in arr[3, 4]

// and place it at beginning of arr[3, 4]

// 11 12 22 25 64

Insertion Sort:- Insertion sort is a simple sorting algorithm that works the way we sort playing cards in our hands.

Algorithm

// sort of arr[] of size n

insertionSort(arr, n)

Loop from $i=1$ to $n-1$

a) Pick element $arr[i]$ and insert it into sorted sequence $arr[0 \text{ to } i-1]$

Example:- 12, 11, 13, 5, 6

Let us loop for $i=1$ (second element of the array) to 6 (last element of array)

$i=1$. Since 11 is smaller than 12, move 12 and insert 11 before 12.

11, 12, 13, 5, 6

$i=2$. 13 will remain at its position as all elements in $A[0 \text{ to } i-1]$ are smaller than 13.

11, 12, 13, 5, 6

$i=3$ 5 will move to the beginning and all other elements from 11 to 13 will move one position ahead of their current position.

5, 11, 12, 13, 6

$i=4$. 6 will move to position after 5, and elements from 11 to 13 will move one position ahead of their current position.

5, 6, 11, 12, 13

4) #include <stdio.h>

void main()

{

int a[100], n, i, j, temp, sum=0, prod=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($i=0$; $i<n$; $i++$)

{


```

for(j=0; j<n; j++)
{
    if(a[j]>a[j+1])
    {
        temp = a[j];
        a[j] = a[j+1];
        a[j+1] = temp;
    }
}

```

```

}
printf("\n Sorted list in ascending order: \n");

```

```

for(i=0; i<n; i++)
{
    printf("%d\n", a[i]);
}

```

```

for(i=0; i<n; 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)
    {
        sum = sum + a[i];
    }
}

```

```

printf("\n sum of odd index is %d", sum);

```

```

for(i=0; i<n; i++)
{
    if(i%2 != 0)
    {
        prod = prod * a[i];
    }
}
}

```

```

for(i=0; i<n; i++)
printf("\n product of odd index & %d", prod);
printf("Enter the value of m");
scanf("%d", &m);
for(i=0; i<n; i++)
{
    if(a[i]*m==0)
    {
        printf("%d", a[i]);
    }
}
}

```

5) #include <stdio.h>

```

int recursiveBinarySearch(int array[], 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 recursiveBinarySearch(array, start-index, middle-1, element);
        return recursiveBinarySearch(array, 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 = recursiveBinarySearch(array, 0, n-1, element);

```

```

    if (found-index == -1) {

```

```

        printf("Element not found");
    }
}

```


che{

printf("Element found at index: %.d", found_index);

}

return 0;

}

②

Program:

#include <stdio.h>

#include <stdlib.h>

// merge two subarrays of arr[]

// First subarray is arr[l to m]

// Second subarray is arr[m+1 to r]

void merge(int arr[], int l, int m, int r)

{

int i, j, k;

int n1 = m - l + 1;

int n2 = r - m;

/* Create temp arrays */

int L[n1], R[n2];

/* Copy data to temp arrays L[] and R[] */

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; // initial index of first subarray

j = 0; // initial index of second subarray

k = l; // initial index of merged subarray

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++;
}

```

```

while (j < n2)
{
    arr[k] = R[j];
    j++;
    k++;
}

```

```

}
}
void mergesort(int arr[], int l, int r)

```

```

{
    if (l < n)
    {
        int m = l + (r - l) / 2;
        mergesort(arr, l, m);
        mergesort(arr, m + 1, r);
        merge(arr, l, m, r);
    }
}
}

```



```
void printArray(int A[], int size)
```

```
{
    int i;
    for(i=0; i<size; i++)
    {
        printf("%d", A[i]);
        printf("\n");
    }
}
```

```
int main()
```

```
{
    int arr[5];
    int i;
    int arr_size = sizeof(arr) / sizeof(arr[0]);
    for(i=0; i<arr_size; i++) {
        printf("Enter element");
        scanf("%d", &arr[i]);
    }
    printf("Given array is\n");
    printArray(arr, arr_size);
    mergeSort(arr, 0, arr_size-1);
    printf("\n Sorted array is\n");
    printArray(arr, arr_size);
    int k;
    printf("Enter value of k");
    scanf("%d", &k);
    int fromFirst = arr[k-1];
    int fromLast = arr[s-(k)];
    printf("%d", fromLast * fromFirst);
    return 0;
}
```