

1) Take the elements from the user and sort them in descending order and do the following

CSE-F

a) Using Binary search from the element and 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 the values at these locations in sorted array.

Program

#include &lt;stdio.h&gt;

void sort (int a[], int n)

{

int i, j, temp;

for (i = 0; i &lt; n; i++)

{

for (j = 0; j &lt; n; j++)

{

if (a[i] &lt; 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 the no. of elements of array:");
```

```
    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]);
```

```
}
```

## Output :

enter the no. of elements of array : 2

enter the elements of array

23

34

3423 is the enter the element to find in array 23

element not found

enter the position of array to find sum and product

2

3

The sum is 23 and The product is 0

2) Sort the array using merge sort where elements are taken from the user and find the product of  $k$ th elements from the list where  $k$  is taken from user.

Program:

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

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

```
// merges two subarrays of arr[]
```

```
// First subarray is arr[l...m]
```

```
// second subarray is arr[m+1...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];
```

/\* merge the temp arrays back into arr[1..8] \*/

i=0; // initial index of first subarray

j=0; // initial index of second subarray

k=1; // 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++;

}

/\* copy the remaining elements of L[], if there are any \*/

while(i<n1)

{

arr[k] = L[i];

i++;

k++;

}

/\* Copy the remaining elements R[], if there are any \*/

while( $i < n_2$ )

{

arr[k] = R[i];

i++;

k++;

}

}

/\* l is for left index and r is right index of the sub-array of  
arr to be sorted \*/

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

{

if ( $l < r$ )

{

// same as  $(l+r)/2$  but avoids overflow for

// large l and h

int m =  $l + (r-l)/2$ ;

// sort first and second halves

merge sort(arr, l, m);

merge sort(arr, m+1, r);

merge(arr, l, m, r);

}

}

/\* UTILITY FUNCTIONS \*/

/\* function to print an array \*/

```

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

```

/\* Driver program to test above functions \*/

```

int main()
{
    int arr[5];
    int i;
    int arr-size = size of (arr) / size of (arr[0]);
    for (i=0; i<arr-size; i++) {
        printf("enter the elements");
        scanf("%d", &arr[i]);
    }
    printf("Given array is \n");
    PrintArray(arr, arr-size);
    merge sort (arr, 0, arr-size-1);
    printf("\n sorted array is \n");
    PrintArray(arr, arr-size);
    int k;
}

```

```
printf("enter the value of k");
```

```
scanf("%d", &k);
```

```
int from first = arr[k-1];
```

```
int from last = arr[5-(k)];
```

```
printf("%d", from last * from first);
```

```
return 0;
```

```
}
```

### Output:

enter the element 2

enter the element 3

enter the element 4

enter the element 5

enter the element 6

Given array is

2 3 4 5 6

Sorted array is

2 3 4 5 6

~~enter~~

enter the value of k is 4

15

3) Discuss Insertion sort and selection sort with examples

### Insertion sort

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

#### Algorithm

// sort an arr[] of size n

insertion sort(arr, n)

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

a) pick element  $arr[i]$  and insert it into sorted sequence  $arr[0 \dots i-1]$

Example: 12, 11, 13, 5, 6

Let us loop for  $i=1$  (second element of the array) to 4 (last element of array)

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

$i=2$  Since 13 will remain at its position as all elements in  $A[0 \dots i-1]$  are smaller than 13

11, 12, 13, 5, 6

$i=3$  5 will move to the beginning and 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

5, 6, 11, 12, 13

### Selection sort

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

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

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

### Example

arr[] = 64 25 12 22 11

// Find the minimum element in arr[0...4]

// and place it on beginning

// 25 12 22 64 11

// Find the minimum element in arr[1...4]

// and place it at beginning of arr[1...4]

// 11 12 25 22 64

// Find the minimum element in arr[2---4]

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

// 12 22 25 64

// Find the minimum element in arr[3---4]

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

// 12 22 25 64

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 positions

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

Program

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

```
void main()
```

```
{
```

```
int a[100], n, i, j, temp, sum=0, prod=1, m;
```

```
printf("Enter the number of elements\n");
```

```
scanf("%d", &n);
```

```
printf("Enter %d integers\n", n);
```

```
for(i=0; i<n; i++)
```

```
{
```

```
    scanf("%d", &a[i]);
```

```
}
```

```
for(j=0; j<n-i-1; 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]);
```

```
}
```

```
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 o = sum o + a[i];  
    }  
}
```

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

```
for(i=0; i<n; i++)
```

```
{  
    if(i%2==0)  
    {  
        prod = prod * a[i];  
    }  
}
```

```
printf("\n product of odd index is %d", prod);
```

```
printf("\n product the value of m/n");
```

```
scanf("%d", &m);
```

```
for(i=0; i<n; i++)
```

```
{  
    if(a[i]%m==0)  
    {  
        printf("%d", a[i]);  
    }  
}
```

## Output :

Enter the number of elements

2

Enter 2 integers

12

13

Sorted list in ascending order

12

13

the alternate order is 12

sum of odd index is 13

Product of odd index is 12

Enter the value of m

5

5) Write a recursive program to implement binary search

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

```
int recursive Binary search( 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 recursive Binary search(array, start-index, middle-1,  
                                           element);
```

```

return recursive Binary search(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 = recursive Binary search(array, 0, n - 1, element);
    if (found_index == -1) {
        printf("Element not found in the array");
    }
    else {
        printf("Element found at index: %d", found_index);
    }
    return 0;
}

```

Output :

Element found at index : 3