

(Chin) 2019

1.

Ans:- #include <stdio.h>  
void sort (int a[], int n)

```
{
    int i, j, temp;
    for (i=0; i<n; i++)
    {
        for (j=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 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[40], j, g, 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);

```

```

    j = binary (a, e, n);

```

```

    if (j != 0)

```



```

    printf("element is found of %d position", j);
}
else {
    printf("Enter the position of array to find sum & product of\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.

```

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

void main (int arr[], int l, int m, int n)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = n - m;
    int l[n1], r[n2];
    for (i = 0; i < n1; i++)
        l[i] = arr[l + i];
    for (i = 0; i < n2; i++)

```

output (1<sup>st</sup> person)

Enter number of elements

Enter 2 integers

12

13

(Sorted list in ascending order)

12

13

the alternate order is 12

sum of odd index = 13

product of odd index = 12

Enter the value of n

5

$R[i] = arr[m+1+j]$

$i=0;$

$j=0;$

$k=1;$

while ( $i < n_1 \ \& \ j < n_2$ )

{  
if ( $L[i] < R[j]$ )

{  
arr[k] = L[i]

$i++;$

}  
else

{ arr[k] = R[j];

$j++;$

}

$k++;$

}

while ( $i < n_1$ )

{ arr[k] = R[j];

$j++;$

$k++;$

}

}



```

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

```

```

void print array (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 = size of (arr) / size of arr[0];

```

```

    for (i=0; i<arr_size; i++) {

```

```

        printf ("enter the element");

```

```

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

```

```

    }

```

```

    printf ("Given array is \n");

```

```

    print array (arr, arr_size);

```

```

    Merge sort (arr, 0, arr_size-1);

```

```

    printf ("\n sorted array is \n");

```

```

    print array (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 1, 2, 3, 4, 5  
 given array  $[ ] = 1, 2, 3, 4, 5$   
 sorted array = 1, 2, 3, 4, 5  
 Enter the value of  $k = 4$   
 $8 = (4 \times 2)$

3.

### Algorithm

insertion sort (arr, n)

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

Example  
 The lower part of an array is maintainance  
 to be sorted An Element which has to be  
 inserted in their sorted But list has to  
 find its appropriate place & then it has  
 to be inserted there



for suppose are need to sort this array

9	5	1	4	3
---	---	---	---	---

5	9	1	4	3
---	---	---	---	---

5	1	9	4	3
---	---	---	---	---

At last

1	3	4	5	9
---	---	---	---	---

### Selection Sort

The Selection Sort algorithm sorts an array by repeatedly finding the minimum element from unsorted part & putting it at the beginning.

### Algorithm

It Minimum two sub arrays in a given array

- 1) The sub array which is already sorted
- 2) Remaining sub array which is unsorted

### Example:-

arr[] = 5 3 4 2 1

Next Step = 1 2 3 4 5

Next Step = 1 2 3 4 5

Next Step = 1 2 3 4 5

Sorted array = 1, 2, 3, 4, 5

4. program

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

```
void main()
```

```
{ int a[100], n, i, temp, sum = 0;
  printf("Enter the no. of elements (n):");
```

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

```
  printf("Enter %d integer (n)", n);
```

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

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

```
}
```

```
for (i = 0; i < n - 1; 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("The array 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("Sum of odd integers is %.d", sum);
```

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

```
{  
    if (i%2 == 0)
```

```
{  
    prod = prod * a[i];
```

```
}
```

```
printf("Product of odd integers is %.d", prod);
```

```
printf("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 Number of elements

2

Enter number of integers

12

13

Sorted list in ascending order

12, 13

Alternate order is 12

Sum of odd integer index is 13

Product of odd index is 12

Enter the value of  $n = 5$

```
5) #include <stdio.h>
int recursive Binary Search (int arr[],
int start_index, int end_index, int element)
{
    if (end_index >= start_index)
    {
        int middle = start_index + (end_index - start_index) / 2;
        if (arr[middle] == element)
            return middle;
        if (arr[middle] > element)
            return recursive Binary Search (arr,
            start_index, middle - 1, element);
    }
}
```



return recursive binary search (array, n-1, end\_index element);

}  
return -1;

{  
int main (void) {

int array[] = {1, 2, 3, 4, 5, 6, 7};

int n = 3

int element = 4

int found\_index = recursive binary search  
(array, n-1, element);

if (found\_index == -1) {  
printf ("Element not found in array");

}  
else {  
printf ("Element found at index %d", found\_index);

}  
return 0;

}

Output :-

Element found at index = 3