Assignment - 6

K.N.V.S.S.NITIN
AP19110010484.
CSE-F.

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

i) Use binary search find the element & the location in the array where the element is asked from user.

ii) Ask the user to print any two locations & find their sum & the product of taken from the user in the sorted array.

Program:-

```
#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 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], 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 & 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) 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 taken from the user.

Program:-

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

void merge(int arr[], int l, int m, int r
{
int i, j, k;
int n1 = m - l + 1;
int n2 = r - m;

int L[n1], R[n2];

for (i = 0; i < n1; i++)
  L[i] = arr[l + i];
for (j = 0; j < n2; j++)
```

Output (1st program)

Enter number of elements : 2.

Enter the 2 integers

12

13

Sorted list is 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[j] = arr[m+1+j];

i = 0;
j = 0;
k = 1;
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] = R[j];
j++;
k++;
}
}
```

```
void mergesort(int arr[], int l, int r)
{
if (l < r)
{

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 the elements");
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 the value of k");
scanf("%d", &k);
int fromfirst = arr[k-1];
int fromlast = arr[5-(k)];

printf("%d", fromlast * fromfirst);

return 0;
}
```

Output

enter the elements - 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 x 2)

3) Discuss insertion sort & selection sort with examples.

⇒ Insertion sort is a simple sorting algorithm that builds the final sorted array one item at a time.

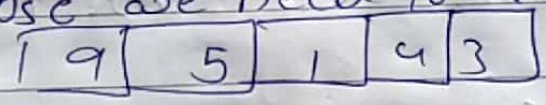
Algorithm

insertion sort (arr, n)
loop from i=1 to n-1.

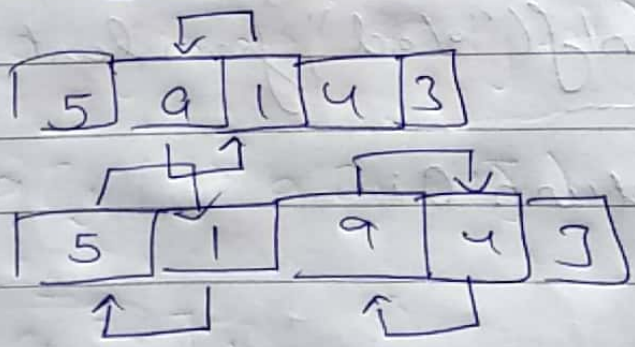
Example

The lower part of an array is maintain
to be sorted. An element which is
to be inserted in this sorted sub-li
has to find its appropriate place
& then it has to be inserted there.

For suppose we need to sort this arra



Step I



At last.

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 maintains two subarrays in a given array

i) The subarray which is already sorted
ii) Remaining subarray which is unsorted

Example

arr [] = 5 3 4 2 1
Next step = 1 3 4 2 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, prod=1, m;
printf("Enter the no. 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-1; i++).
{
for(j=0; j<n-1-1; j++).
{
  if(a[j] > a[j+1]).
}
{
temp = a[j];
  a[j] = a[j+1];
   a[j+1] = temp;
  }
 }
}
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)
{
  sumO = sumO + 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 number of elements
2
Enter number of integers
12
13
Sorted list in ascending order
12 ; 13
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.

Program

```
#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(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, 2, 3, 4, 5, 6, 7};
int n = 3.
int element = 4.

int found_index = recursive binary -
                search(array 0, 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