C ASSIGNMENT

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

a) Use Binary Search find the element and the location in array where the element is asked from user.

A)

```
#include<stdio.h>
void main(){
int c, first, last, middle, n, search, array[100];
Printf("Enter number of elements\n");
scanf("%d",&n);
printf("Enter integers %d\n", n);
for(c=0; c<n; c++)
   scanf("%d", &array[c]);
printf("Enter a value to find\n");
scanf("%d", &search);
first=0;
last=n-1;
middle = (first+last)/2
while (first <= last){
if(array[middle] < search)
   first = middle+1;
else if(array[middle]== search){
   printf("%d found at location %d.\n", search, middle+1);
   break;
}
else
last= middle-1;
```

```
middle = (first + last)/2;
}
if (first > last)
printf("%d isn't present in the list %d.\n", search);
}
```

Output:-

Enter number of elements = 3

Enter 3 integers = 50
8
2

Enter value to find = 8

8 found at location 2.

b) Ask the user to enter any two locations print the sum and product of values at those locations in the sorted array.

A)

```
#include <stdio.h>
int void main(){
int L1, L2, a[10];
for(int i=0; i<=10; i++) {
    printf("Enter any two locations in array");
    scanf("%d", &L1);
    scanf("%d", &L2);
    sum = a[L1] + a[L2];
    product = a[L1] * a[L2];
    printf("Sum is %d", sum);
    printf("product is %d", product);
    printf("Enter 10 elements in the array");
    printf("Enter the values to be found in location");
}
```

Output:-

Enter elements in the array: 3

2
5
10

Enter values to be found in location: 2
10

Sum is 12

Product is 20.

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

A)

```
#include<stdio.h>
#include<stdlib.h>
void merge (int arr[], int p, int q, 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++)
R[j] = arr[m+1+j];
i=0;
j=0;
k=l;
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<r)
{
int m = l + (r-l)/2;
mergesort (arr, l, m);
mergeSort (arr, m+1, r);
merge (arr, l, m, r);
}
}
int main()
{
 int a[100], n, k;
printf ("Enter number of elements");
scanf ("%d", &n);
for (int i=0; i<n; i++){
printf ("Enter next element");
scanf ("%d", &a[i]);
}
mergeSort (0, n-1, a);
printf("Sorted array");
 for (int i=0; i<n; i++)
printf ("%d", a[i]);
printf (" Enter k value to find product of kthelement:");
Scanf ("%d", &k);
printf (" The product is: %d", a[k-1]*a[n-k];
return 0;
}
```

3) Discuss insertion sort and Selection sort with examples.

A)

Insertion sort	Selection sort
1) Insertion sort is a simple algorithm that builds the final sorted array.	1) Selection sort finds the minimum number from the list.
2) Insertion sort removes one element from the input data.	2) The data is sorted by selecting the consecutive elements in sorted location
3) It inserts the next element at the correct position.	3) Location is previously known while elements are searched
4) Insertion sort inserts the values in a presorted file.	4) It can not deal with immediate data it needs to be present at the beginning.
5) Best-case Complexity is $O(n)$	5) Best case complexity is $O(n^2)$
Eg:- The lower part is maintained to be sorted. An element which is to be inserted has to find a appropriate place.	Eg:- Sort the elements using an array,

4) Sort the array using bubble sort where elements are taken and display the elements.

1) alternate Order

2) Sum of elements in odd positions and Product of elements in even positions.

3) Elements which are divisible by m

A)

```
#include <stdio.h>
void bubbleSort (int a[], int n)
{
int i,j, temp;
for (i=0; i<n-1; i++)
for (j=0; j<n-1-i; j++)
if (a[j] > a[j+1]){
temp= a[j];
a[j]=a[j+1];
a[j+1] = temp;
}
}
int main()
{
int a[100], n, sum=0, product=1, m, f=0;
printf("Enter the number of elements:");
scanf("%d", &n);
for (int i=0; i<n; i++){
printf("Enter next element:");
scanf("%d", &a[i]);
}
bubble sort (a,n);
```

```
printf(" Sorted array: \n");
for (int i=0; i<n; i++)
printf("%d", &a[i])
printf(" Elements in alternating order:\n");
for (int i=0; i<n; i+=2)
printf(" %d ", a[i]);
printf(" Elements in alternating order \n");
for (int i=0; i<n; i++) {
if (i%2==0)
sum+= a[i];
else
product*= a[i];
}
printf(" Sum of elements in odd position: %d\n", sum);
printf(" Product of elements in even position: %d", product);
printf(" \n Enter the value of m to check divisibility");
scanf(" %d ", &m);
for (int i=0; i<n; i++) {
if (a[i]%m==0) {
printf(" %d ", a[i]);
f++;
}
else
continue;
}
if(f==0)
printf(" No elements divisible by m");
return 0;
}
```

5) Recursive Program to implement Binary Search.

A)
```
#include <stdio.h>
int binarySearch (int arr[], int l, int r, int x)
{
if (r >= l) {
int mid = l + (r - l)/2 ;
if (arr[mid] == x)
return mid;
if (arr[mid] > x)
return binarySearch (arr, l, mid-1, x);
return binarySearch (arr, mid+1, r, x);
}
return -1;
}
int main(void)
{
 int a[100], n, r, x;
 printf(" Enter the elements in ascending order \n");
 scanf("%d", &n);
 for(int i =0; i<n; i++){
 printf("Enter next element:") ;
 scanf("%d",&a[i]);
}
printf("Enter element to be searched") ;
scanf ("%d", &x);
int result = binarySearch (a,0, n-1, x);
(result == -1)
 printf(" Element is not present in array \n"); printf("Element is at index
 %d \n", result)
}
```