ASSIGNMENT - 6

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

a) Using Binary search find the element and the 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 of values at those locations in the sorted array.

Ans)

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

int comparator (const void *a , const void *b){
    return (*(int*)b - *(int*)a);
}
int binarySearch (int arr[], int size , int search){
    int top =1 , end = size -1 , mid;
    while (top <= end){
        mid = (top + end)/2;
        if (arr[mid] = = search){
            return mid;
        }
        else if (arr [mid] < search) {
            end = mid -1;
        else top = mid+ 1;
}
```

```
    return -1;
}
int main(){
    int arr[100], size, search, i, pos = -1, loc1, loc2;
    printf("\n Size of array (max 100):");
    scanf("%d", &size);
    printf("\n Enter elements : \n");
    for (i=0; i< size; i++){
        scanf("%d", &arr[i]);
    }
    qsort(arr, size, size of (int), comparator);
    printf("\n The stored array is : \n");
    for (i=0; i<size; i++)
        printf("%d", arr[i]);

    printf("\n Enter element to search : ");
    scanf("%d", &search);
    pos = binarySearch(arr, size, search);
    if (pos == -1) printf("Not found !!! ");
    else printf("\n The searched element %d is found at
                the position %d \n", search, pos);
    printf("Enter two indices \n");
    scanf("%d %d, &loc1, &loc2);
printf("sum is %d \n", arr[loc1] + arr[loc2]);
printf("product is %d \n", arr[loc1] * arr[loc2]);
    return 0;
}
```

Output :

Size of array (max 100): 6

Enter elements :

4 7 2 9 3 5

The sorted array is :

9 7 5 4 3 2

Enter element to search : 5

The searched element 5 is found at the position 2

Enter two indices

4 5

sum is 5

product is 6

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.

Ans)

```
#include <stdio.h>
# define max 100
int l[max];
void merge (int a1, int b1, int a2, int b2){
    int i, j, k, temp[max];
```

```
k = 0;
i = a1;
j = a2;
while((i <= b1) && (j <= b2)) {
    if (l[i] < l[j]){
        temp[k] = l[j]; i++; k++;
    }
    else {
        temp[k] = l[i]; j++; k++;
    }
}
while (i <= a1){
    temp[k] = l[i]; i++; k++;
}
while (j <= a2){
    temp[k] = l[j]; j++; k++;
}
for (i = a1, k=0; a<b2; i++, k++){
    l[i] = temp[k];
}
}
void mergesort (int ag, int bg){
    if (ag <bg){
        int mid = (ag + bg)/2;
        mergesort (ag, mid);
        mergesort (mid + 1, bg);
```

```
        merge(aq, mid, mid+1, bg);
    }
}
int main(){
    int i, n, product=1, k;
    printf("\n Size of array (max 100):");
    scanf("%d", &n);
    for (i=0; i<n; i++){
        printf("l[%d] \t = ", i);
        scanf("%d", &l[i]);
    }
    mergesort(0, n-1);
    printf("Enter k \n");
    scanf("%d", &k);
    for (i=0; i<k; i++){
        product k = l[i];
    }
    printf("\n The product till the kth element is
            %d \n", product);
    return 0;
}
```

OUTPUT:

Size of array (max 100): 6

l[0] = 5 l[2] = 3 l[4] = 8 Enter k

l[1] = 2 l[3] = 1 l[5] = 1 4

The product till k^{th} term is 6.

3) Discuss insertion sort and selection sort with examples.

Ans)

Insertion sort :

Suppose an array A with n elements A[1], A[2], A[3] ..... A[n] is in memory, the insertion sort algorithm scans A from A[1] to A[n], insertion of each element A[k] into its proper position in the previous sorted sub array [A[1], A(2), A(3) .... A(k-1)].

Example :

initially : 20 30 1 4 13

~~pa~~ 1st → 20 30 1 4 13

2nd → 1 20 30 4 13

3rd → 1 20 30 4 13

4th → 1 4 20 30 13

5th → 1 4 13 20 30 sorted

Pseudocode :

```
1) for j = 2 to A.length
      key = A[j]
      // Insert A[j] into the sorted sequence A[1 ... j-1]
      i = j - 1
      while (i > 0 and A[i] > key)
          A[i+1] = A[i]
```

$i = i - 1$

$A[i+1] = key$

Time complexity :

Best : $O(n)$

Worst : $O(n^2)$

Average : $O(n^2)$

Space complexity :

$O(1)$

Selection sort :

The basic idea of selection sort is repeatedly select the smallest key in the unsorted array.

Example :

		20	30	1 → smallest	8	15
1st	1	20	30	8 → smallest	15
2nd	1, 8	20	30	15 → smallest	
3rd	1, 8, 15	20 → smallest	30		
4th	1, 8, 15, 20	30 → smallest			
5th	1, 8, 15, 20, 30 → sorted				

Pseudo code :

```
n ← length[A]
for j ← 1 to n-1
    do smallest ← j
        for i ← j+1 to n
            do if A[i] < A[smallest]
                then smallest ← i
        exchange A[j] ↔ A[smallest]
```

Time complexity :
Best : $O(n)$
Worst : $O(n^2)$
Average : $O(n^2)$

Space complexity :
$O(1)$

4) Sort the array using bubble sort where elements are taken from user and display the elements.
i) in alternate order
ii) sum of elements in odd positions and product of elements is even positions
(iii) Elements which are divisible by m where m is taken from user.

Ans)

```
#include <stdio.h>
void displayAltSum(int arr[], int size){
    int i, sum=0, product=1;
    printf("Alternate elements\n");
    for(i=0; i<size; i++){
        if(i%2 !=0){
            product += arr[i];
        }
        else {
            sum += arr[i];
            printf("%d", arr[i]);
        }
}
```

```
printf("\n Sum of the odd elements = %d \n", sum);
print f("\n Sum of the even elements = %d \n", product);

void divM(int arr[], int size){
    int i=0, m;
    print f("Enter the m \n");
    scanf("%d ", &m);
    print f("Elements divisible by %d \n", m);
    for (i=0; i<size; i++){
        if (arr[i]% m ==0)
            print f("%d ", arr[i]);
    }
}
void bubblesort (int arr[], int size)
{
    int i, j, temp;
    for (i=0; i<size-1; i++)
        for (j=0; j<size-i-1; j++)
            if (arr[j] > arr[j+1]){
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }

    display AltSumPro(arr, size);
    divM(arr, size);
}
```

```
int main ()
{
    int arr [100], size, i;
    printf("/n Enter the size of the array (max 100)");
    scanf ("%d", & size);
    printf("\n Enter elements in array \n");
    for (i=0; i<size; i++){
        scanf ("%d", & arr [i]);
    }
    bubble Sort (arr, size-1);
    return 0;
}
```

Output :

Enter the size of the array (max 100) 6

Enter the elements in array

15

2

7

8

10

1

Alternate elements

2 7

Sum of the odd elements = 18

Product of the even elements = 240

enter the m

5

Elements divisible by 5

10, 15

5) Write a recursive program to implement binary search ?

Ans)

```
#include <stdio.h>
int binarySearch (int arr[], int top, int end, int search)
    int mid;
    if (top <= end) {
        mid = (top + end)/2;
        if (arr[mid] == search) return mid;
        if (arr[mid] > search);
            return binarySearch (arr, top, mid-1,
                                search);
        return binarySearch (arr, mid+1, end,
                                search);
    }
    return -1;
}
int main() {
    int arr[100], size, search, i, pos;
    printf("\n Size of the array (max 100)");
    scanf("%d", &size);
    printf("\n Enter sorted elements in array\n");
    for (i=0; i<size; i++) {
        scanf("%d", &arr[i]);
}
```

```
    printf("\n Enter searched element");
    scanf("%d", &search);
    pos = binarySearch(arr, 0, size-1, search);
    if (pos == -1)
        printf("Not found !!! \n");
    else
        printf("\n The searched element %d is found
                at index %d \n", search, pos);

    return 0;
}
```

OUTPUT :

Enter the size of array (max 100) 5

Enter sorted elements in array

15 20 25 30 35

Enter search element 25

The searched element 25 is found at index 2.