DSA-ASSIGNMENT

M. Hartha
AP19110010270

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

a) Using Binary search find the element 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.

```
#include <stdlib.h>
#include <stdio.h>
int comparator(const void *B1, const void *B2){
    return (*(int*)B2 - *(int*)B1);
}
int binarysearch(int arr[], int size, int search)
{
    int beg=0, end=size-1, mid;
    while (beg<=end){
        mid=(beg+end)/2;
        if (arr[mid]==search){
            return mid;
        }
        else if (arr[mid]<search){
            end=mid-1;
```

```
        else beg = mid+1;
    }
    return -1;
}
int main(){
    int arr[100], size, search, i, pos=-1, loc1, loc2;
    printf("\n Enter the size of the array (max 100)");
    scanf("%d", &size);
    printf("\n Enter the elements \n");
    for(i=0; i<size; i++){
        scanf("%d", &arr[i]);
    }
qsort(arr, size, sizeof(int), comparator);
printf("\n the sorted array is :\n");
for(i=0; i<size; i++)
    printf("%d", arr[i]);
printf("\n Enter search element");
scanf("%d", &search);
Pos = binarysearch(arr, size, search);
if(Pos==-1){
    printf("Not found \n");
}
```

```
else
{ printf("\n the %d search element is found
        at index %d \n", search, Pos);
}
printf("Enter two indexes\n");
scanf("%d%d", &loc1, &loc2);
printf("sum is %d\n", arr[loc1]+arr[loc2]);
printf("product is %d \n", arr[loc1]*arr[loc2]);
}
```

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.

```
#include<stdio.h>
#define mx 100
int a[mx];
void merge(int l1, int u1, int l2, int u2)
{
  int i,j,k,temp[mx];
  k=0;
  i=l1;
  j=l2;
  while((i<=u1) && (j<=u2)){
    if(a[i]<a[j]){
```

```
        temp[k] = a[i]; i++; k++;
    }
    else
    {
        temp[k] = a[j]; j++; k++;
    }
}
while (i<=u1) {
    temp[k] = a[i]; i++; k++;
}
while (j<=u2) {
    temp[k] = a[j]; j++; k++;
}
for (i=l1, k=0; i<=u2; i++, k++) {
    a[i] = temp[k];
}
}
void mergesort (int lb, int ub) {
    if (lb<ub)
    {
        int mid = (ub+lb)/2;
        mergesort (lb, mid);
        mergesort (mid+1, ub);
        merge (lb, mid, mid+1, ub);
    }
}
```

```
int main()
{ int i,n,product=1,k;
  printf("\n Enter the size of the array, max(100)");
  scanf("%d",&n);
  for(i=0;i<n;i++){
     printf("a[%d]\t=",i);
     scanf("%d",&a[i]);
}.
mergesort(0,n-1);
printf("Enter k\n");
scanf("%d",&k);
for(i=0;i<k;i++){
   product*=a[i];
}
Printf("\n the product till the kth element is %d\n",
        product);
return 0;
}
```

(3) Discuss Insertion sort and selection sort with examples.

Insertion sort

It is a simple and efficient sorting algorithm, that creates the final sorted array one element at a time. Suppose an array A with n elements

then the insertion sort algorithm scans A from A[1] to A[N], insertion each element A[K] into its proper position (i.e in Ascending or descending order) in the previous sorted sub array A[1], A[2] --- A[k-

Example:-

Given array: 20, 25, 15, 11, 9

Pass 1: 20, 25, 15, 11, 9

~~20 15, 25, 11 9~~

Pass 2: 15, 20, 25, 11, 9 Swap 20, 15

Pass 3: 11, 15, 20, 25, 9 Swap 20, 15, 11

Pass 4: 9, 11, 15, 20, 25 Swap 9

sorted.

Pseudocode

Procedure A: array of items

int key

for i ← 2 to length[A]

do key ← A[i]

Insert A[i] into the sorted sequence A[1 - - - i-1]

j ← i-1

while j>0 and A[j] > key

do A[j+1] ← A[j]

j ← j-1

A[j+1] ← key

end while

end for

end procedure

time complexity:

best: O(n) average: O(n²) worst O(n²)

space complexity: O(1)

selection sort:

It is like selecting the smallest key repeatedly in the unsorted array to make sorted array.

Example:- 16, 19, 10, 6, ⑤ - smallest

Pass 1: 5 16, 19, 10, ⑥ - smallest

Pass 2: 5, 6 16, 19, ⑩ - smallest

Pass 3: 5, 6, 10, 16, 19 smallest

Pass 4: 5, 6, 10, 16, 19

Pass 5: 5, 6, 10, 16, 19 - sorted.

Pseudo code

small = arr[length]

for i = 2 to

small = arr[i], pos = j

for j = i to do {

if arr[j] < small then {

small = arr[i], pos = (i)

}

j = j+1

temp = arr[i], arr[i] = small, arr(pos) = temp)

}

END.

Time complexity:

best: o(n) average: o(n^2) worst: o(n^2)

space complexity: o(1)

4. Sort the array using bubble sort where elements are taken form 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 the user.

```
#include <stdio.h>
void displayAltSumPro(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(" In sum of the odd elements = %d \n", sum);
Printf(" In product of the even elements = %d \n", product);
}.
void divm(int arr[], int size){
    int i=0, m;
    Printf ("Enter the m \n");
    scanf ("%d", &m);
    printf(" elements divisible by %d \n", m);
    for (i=0; i<size; i++) {
        if (arr[i] % m==0)
            print+("%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;
            }.
    displayAltSumPro(arr, size);
    divm(arr, size);
```

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

5. Write a recursive program to implement binary search

```
#include <stdio.h>
int binarysearch(int arr[], int start, int end, int search){
        int mid;
        int(start<=end){
            mid = (start+end)/2;
            if(arr[mid] == search) return mid;
            if (arr[mid]>search){
                return binarysearch(arr, start, mid-1,
                                    search);
            };
            return binarysearch(arr, mid+1, end, search
        }.
        return -1;
```

```
int main()
{
    int arr[100], size, search, i, pos;
    printf("\n Enter the 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 search element");
    scanf("%d", &search);
    pos = binarysearch(arr, 0, size-1, search);
    if(pos==-1) printf("Not found\n");
    else printf("\n the %d search element is found at
                index %d\n", search, pos);
    return 0;
}.
```