DSA Assignment

SK. Kishwar Parveen
AP19110010545
CSE-G

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.

```
# include<stdio.h>
# include <stdlib.h>
int comparator (const void *p1, const void *p2)
{
    return (*(int*)p2 - *(int*)p1);
}
int Binary Search (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[101], size, search, i, pos=-1, loc1, loc2;
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]);
}
```

```
qsort (arr, size, size of (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 = binary Search (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]);
}
```

OUTPUT: (1)

Enter the size of The array (max 100) 5

Enter elements in array

5 2 3 6 7

The sorted array is

7 6 5 3 2

Enter search element 2

The 2 search element is found at index 4

Enter two indexes

2 3

Sum is 8

product = 15

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

```
#include<stdlib.h>
#include <stdio.h>
void merge (int arr[], int x, int y, int z)
{
int i, j, k;
int n1 = y - x + 1;
int n2 = z - y;
int X[n1], Z [n2];
for (i = 0 ; i < n1 ; i++)
X [i] = arr [x + i];
for (j = 0; j < n2 ; j++)
Y[j] = arr [y + 1 + j];
i = 0
j = 0
k = 1
```

```
while (i<n1 && j<n2)
{
if (X[i]<=Z[j])
{
arr[K]=X[i];
i++;
}
else
{
arr[K]=Z[j];
j++;
}
K++;
}
while (i<n1)
{
arr[K]=X[i];
i++;
K++;
}
while (j<n2)
{
arr[K]=Z[j];
```

```
j++;
K++;
}
}
void merge sort (int arr[], int X, int Z)
{
if (X<Z)
{
int m= X+(Z-X)/2;
merge sort (arr, X, m);
merge sort (arr, m+1, Z);
merge (arr, X, y, Z);
}
}
int main()
{
int a[100], n, k;
printf ("Enter the number of elements");
Scanf ("%d", &n);
for (int i= 0, i<n, i++){
```

```
printf ("Enter next element");
scanf ("%d", &a[i]);
}
mergesort (a, 0, n-1);
printf ("Sorted Array:");
for (int i=0; i<n; i++)
printf ("%d", a[i]);
printf ("\n Enter k value to find the product
of Kth element from first & last:");
scanf ("%d", &k);
printf ("The product is: %d", a[k-1] * a[n-k]);
return 0;
}
```

3) ~~Discuss~~

OUTPUT:

```
Enter the size of the array 5
a[0] = 1
a[1] = 6
a[2] = 1
a[3] = 54
```

a[4] = 2

Enter K

3

The product till the Kth element is 2

3) Discuss Insertion Sort & Selection sort with examples

Sol: Insertion Sort:

Insertion sort works by inserting the set of values in the existing sorted file. It consists the sorted array by inserting a single element at a time. This process continues until whole array is sorted in the same order. The primary concept behind insertion sort is to insert each item into its appreciate place in the final list. The insertion sort method saves an effective of memory.

Working of Insertion Sort:

Suppose an array A with n elements A[1], A[2], ---- A[N] is in memory. This insertion sort algorith scans A from A[1] to A[N], insertion each element A[K] into its proper position in the previous sorted sub array A[1], A[2] ... A[K-1]

Example:

array initial : 15, 19, 12, 21, 9

pass 1 : 15, 19, 12, 21, 9

pass 2 : 12, 15, 19, 21, 9

pass 3 : 12, 15, 19, 21, 9

pass 4 : 9, 12, 15, 19, 21

sorted

Pseudo Code:

1) A[10] = minimum integer value

2) Repeat steps 3 through 8 for

K = 1, 2, 3, ..., N-1

{

3) temp = A[K]

4) ptr = k-1

5) Repeat steps 6 to 7 while temp < A[ptr]

{

6) A[ptr+1] = A[ptr]

7) ptr = ptr-1

}

8) A[ptr+1] = temp

}

9) END

<u>time complexity</u>

Best: $O(n)$. average $O(n^2)$ worst $(O(n^2))$

space complexity : $O(1)$

Selection sort:

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

Example: 15, 6, 13, 3, 2 → smallest

Pass 1: 2 15, 6, 13, 3 smallest

pass 2: 2, 3 15, 6, 13 smallest

Pass 3: 2, 3, 6 15, 13 smallest

Pass 4: 2, 3, 6, 13 15 smallest

pass 5: 2, 3, 6, 13, 15 → sorted

Pseudo code:

```
1) small
2) FOR i = 2 to n do {
3) small = AR[i], pos[]
4 FOR j = i+1 U do{
5 IF AR[j] < small then{
6
7 }   small = AR[i] , pos = [i]
8 j = j+1
temp = AR[i], AR[i] = small.
9 }   AR[pos] = temp )
END
```

Time complexity

Best : O(n)
average (on²)
worst O(n²)

Space complexity
O(1)

4) Sort the array using bubble sort where elements are taken from the user and display 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("\n Sum of the odd elements
                    %d\n", sum);

    printf("\n product of the even element
                    = %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)
            printf(" %d ", arr[i]);
    }

}
void bubble sort (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 Alt SumPro (arr, size);
    div M (arr, size);
}
int main ()
{
        int arr[100], size, i;
        printf ("\n Enter the size of the array
                    (max 100)");
        scanf ("%d ", &size);
        print f ("\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) 5

Enter elements in array

9 4 5 2 8

Alternate elements

2 5

Sum of the odd elements = 7

product of the even elements = 14

Enter the m

3

Elements divisible by 3

9

5) Write a recursive program to implement binary search?

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

```
}
int main()
{
    int arr[100], size, search, i, pos;
    printf("\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 = binary search (arr, 0, size-1, search);
```

```
found at index %d\n", search,pos);
return 0;
}
```

OUTPUT:

```
Enter the size of the array (max 100)5
Enter sorted elements in array
1
2
3
4
5
Enter search element 3
the 3 search element is found at
index 2
```