DSA Assignment - 6
Searching and Sorting

P. Kundhana Harshitha
AP19110010377
CSE-H

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 the user.

```
#include<stdio.h>
void main(){
int array[100], n, c, d, temp, first, last, middle search;
printf("Enter number of elements : ");
scanf("%d", &n);
printf("Enter %d integers : ", n);
for(c=0; c<n; c++){
    scanf("%d", &array[c]);
} printf("Enter the element to search: ");
 scanf("%d", &search);
 for(c=0; c<n-1; c++){
 for(d=0; d<n-c-1; d++){
        if(array[d] < array[d+1]){
            temp = array[d];
            array[d] = array[d+1];
            array[d+1] = temp;

}}}
 printf("Sorted list in descending order: \n");
 for(c=0; c<n; c++){
        printf("%d\t", array[c]);
}
```

```
first = 0;
last = n-1;
while (first <= last){
middle = (first + last) / 2;
if (array[middle] < search){
        last = middle - 1;
}else if (array[middle] > search){
        first = middle + 1;
}else if (array[middle] == search){
        printf("\n%d is found at position %d", search, middle+1);
        break;
    }
}
if (first > last){
        printf("%d is not found", search);
}
}
```

Output:

```
Enter number of elements : 5
Enter 5 integers : 1 9 8 4 6
Enter the element to search: 8
Sorted list in descending order:
9   8   6   4   1
8 is found at position 2
```

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

```
#include<stdio.h>
void main(){
int array[100],n,c,d,temp,loc1,loc2,sum,prod;
printf("Enter number of elements:");
scanf("%d",&n);
printf("Enter %d integers :",n);
for(c=0;c<n;c++){
    scanf("%d",&array[c]);
}
for(c=0;c<n-1;c++){
for(d=0;d<n-c-1;d++){
      if(array[d]<array[d+1]){
            temp=array[d];
           array[d]=array[d+1];
           array[d+1]=temp;
}}}
printf("Sorted list in descending order:\n");
for(c=0;c<n;c++){
    printf("%d\t",array[c];
}
printf("\n Enter any location below %d :",n+1);
scanf("%d",&loc1);
printf("\n Enter any location below %d :",n+1);
scanf("%d",&loc2);
sum=array[loc1-1]+array[loc2-1];
prod=array[loc1-1]*array[loc2-1];
printf("\nSum of elements at locations %d and %d
      is %d",loc1,loc2,sum);
```

```
printf("\n Product of elements at locations %d and
      %d is %d", loc1, loc2, prod);

}
```

Output:

```
Enter number of elements: 5
Enter 5 integers : 5 1 6 4 8
Sorted list in descending order:
8    6    5    4    1
Enter any location below 6: 2
Enter any location below 6: 3
Sum of the elements at locations 2 and 3 is 11
Product of the elements at locations 2 and 3 is 30
```

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 user.

```
# include <stdio.h>
void mergesort (int a[], int i, int j);
void merge (int a[], int i1, int j1, int i2, int j2);
int main(){
    int a[30], n, i, k, prod=1;
    printf("Enter no of elements: ");
    scanf("%d", &n);
    printf("Enter array elements: ");
    for(i=0; i<n; i++){
```

```
        scanf ("%d", &a[i]);
    }
    printf ("\n Enter the kth value : ");
    scanf ("%d", &k);
    mergesort (a, 0, n-1);
    printf ("\n sorted array is : ");
    for (i=0; i<n; i++){
        printf ("%d", a[i]);
    }
    for (i=0; i<n; i++){
        if (i==k-1 || i==n-k){
            prod = prod * a[i];
        }
    }
    printf ("\n Product is %d ", prod);
    return 0;
}
void mergesort (int a[], int i, int j){
    int mid;
    if (j>i){
        mid = (i+j)/2;
        mergesort (a, i, mid);
        mergesort (a, mid+1, j);
        mergesort (a, i, mid, mid+1, j);
    }
}
void merge (int a[], int i1, int j1, int i2, int j2){
```

```
int temp[50];
int i,j,k;
i=i1;
j=i2;
k=0
while(i<=j1 && j<=j2){
    if(a[i]<a[j]){
        temp[k++]=a[i++];
    }else{
        temp[k++]=a[j++];
    }
}
while(i<=j1){
    temp[k++]=a[i++];
}while(j<=j2){
    temp[k++]=a[j++];
}for(i=i1,j=0;i<=j2;i++,j++){
    a[i]=temp[j];
}
}
```

Output:-

```
Enter no of elements: 6
Enter array elements: 9 2 8 3 6 7
Enter the kth value: 3
sorted array is: 2 3 6 7 8 9
Product is 42
```

3) Discuss Insertion sort and Selection sort with examples.

Selection Sort:

Introduction:

This is a simple sorting algorithm. In this sorting the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Finally whole of the list is sorted.

Code:

```
# include <stdio.h>
int main (){
    int i,j,n, array[100], position, temp;
    printf ("Enter the number of elements: ");
    scanf ("%d", &n);
    printf (" Enter %d elements : ", n);
    for (i=0; i<n; i++){
        scanf ("%d", &array[i]);

    }
    for (i=0; i<n; i++){
        position = i;
        for (j= i+1; j<n; j++){
            if (array[position] > array[j]) {
                position = j;

            }
    } if (position != i){
            temp = array[i];
            array[i] = array[position];
            array[position] = temp;
}}
```

```
    printf("Elements are sorted in ascending order:");
    for(i=0; i<n; i++){
        printf("%d\t", array[i]);
    }
}
```

Example:

Consider the array: [1, 9, 3, 7, 5]

Let us run the loop for i=1 (second element of the array) to 4 (last element of the array).

i=1

The first element is 1. Now we must find the smallest number from the remaining array than 1. The smallest no from 9 3 7 5 is 3 but is not smaller than 1 so the number in first place is not replaced by any number.

i=2

Now the second element is 9. Finding the smallest element from the remaining elements 3 7 5 is 3.

9 is replaced with 3. The array is [1, 3, 9, 7, 5].

i=3

Now the third element is 9. Finding the smallest element from the remaining elements 7 5 is 5.

So 9 is replaced with 5. The array is [1, 3, 5, 7, 9]

i=4

Now, the fourth element is 7 which is smaller than 9. All the elements are sorted.

Finally, sorted array is [1, 3, 5, 7, 9].

Insertion Sort:

Introduction:-

The strategy behind the insertion sort is similar to the process of sorting a pack of cards. You can take a card, move it to its position location in sequence and move the remaining cards left or right as needed.

Code:

```
# include <stdio.h>
int main(){
    int n, i, array[100], pos, temp;
    printf("Enter number of elements: ");
    scanf("%d", &n);
    printf("Enter %d number of elements : ", n);
    for(i=0; i<n; i++){
        scanf("%d", &array[i]);
    } for(i=1; i<n; i++){
        pos=i
        while(pos>0 && array[pos-1]>array[pos]){
            temp = array[pos-1];
            array[pos-1] = array[pos];
            array[pos] = temp;
            pos--;
        }
    } printf("Elements are sorted in ascending order:");
    for(i=0; i<n; i++){
        printf("%d\t", array[i]);
}}
```

Example:

9, 5, 8, 2, 6

for i=1 (2nd element) to 4 (last element)

i=1

Since 5 is smaller than 9, move 9 and insert 5 before 9.

5, 9, 8, 2, 6

i=2

Since 8 is smaller than 9, move 9 and insert 8 before 9.

5, 8, 9, 2, 6

i=3

2 will move to the beginning and all other elements from 5 to 9 will move one position ahead of their current position.

2, 5, 8, 9, 6

i=4. 6 will move to position after 5, and elements from 8 to 9 will move one position ahead of their current position.

Finally the sorted list is: 2, 5, 6, 8, 9

4) Sort the array using bubble sort where elements are taken from 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>
int main()
int i,j,temp,n, array [100], sum=0, prod=1, m=, count=0;
printf ("Enter number of elements in the array: ");
scanf ("%d", &n);
printf ("Enter elements: ");
for (i=0; i<n; i++){
    scanf ("%d", &array[i]);
}
printf ("Enter the value of m: ");
scanf ("%d", &m);
for (i=0; i<=n-1; i++){
    for (j=0; j<n-i-1; j++){
        if (array[j] > array[j+1]){
            temp = array[j+1];
            array[j+1] = array[j];
            array[j] = temp;
        }
    }
}
printf ("Elements are sorted in ascending order: ");
for (i=0; i<n; i++){
    printf ("%d", array[i]);
}printf ("\n Elements in alternate order are : ");
for (i=0; i<n; i=i+2){
    printf ("%d", array[i]);
}for (i=0; i<n; i++){
    if (i%2==0){
        sumt = array[i];
}}
```

```
printf("\n Sum of elements in odd positions is %d", sum);
for(i=0; i<n; i++){
        if (i%2!=0){
            prod = prod * array[i];
        }
}
printf("\n Product of elements in even positions is %d", prod);
printf("\n The numbers that are divisible by %d are: ", m);
for(i=0; i<n; i++){
        if (array[i]%m==0){
            printf("%d", array[i]);
            count++;
        }
} if (count==0){
        printf("none");
}}
```

Output:

```
Enter number of elements in the array: 6
Enter elements: 1 8 3 5 7 2
Enter the value of m: 2
Elements are sorted in ascending order: 1 2 3 5 7 8
Elements in alternate order are: 1 3 7
Sum of elements in odd positions is 11
Product of elements in even positions is 80
The numbers that are divisible by 2 are: 2 8
```

5) Write a recursive program to implement binary search?

```
# include <stdio.h>
void  binary_search (int [], int, int, int);
void  bubble_sort (int [], int);
int  main(){
     int  key, i, a[100];
     printf ("enter size of array: ");
     scanf ("%d", &n);
     printf ("enter elements: ");
     for(i=0 ; i<n; i++){
          scanf ("%d", & a[i]);
     }
bubble_sort(a, n);
printf ("\n");
printf ("enter  key  to  search : ");
scanf ("%d ", & key);
binary_search (a, 0, n, key );
}
void  bubblesort (int  a[], int n){
     int  temp, i, j ;
     for(i=0; i<n; i++){
        for(j=i ; j<n; j++){
           if (a[i] >a[j]){
              temp = a[i];
              a[i] = a[j];
              a[j] = temp;
           }
        }
     }
printf ("Sorted  array  is : ");
for(i=0 ; i<n; i++){
     printf ("%d", a[i]);
} }
```

```
void binary_search (int a[], int f, int l, int key) {
    int mid;
    if (f > l){
        printf ("%d is not found at any position \n", key);
        return;
    }
    mid = (f + l)/2;
    if (a[mid] == key){
        printf ("%d is found at position %d \n", key, mid+1);
    }else if (a[mid] > key){
        binary_search (a, f, mid-1, key);
    }else if (a[mid] < key){
        binary_search (a, mid+1, l, key);
    }
}
```

Output:

```
Enter the size of the array: 5
Enter elements: 1 8 9 5 4
Sorted array is: 1 4 5 8 9
Enter key to search: 5
5 is found at position 3
```