ASSIGNMENT - 6

SHREYA PANDRANGI
AP19110010433
CSE - H

Program 1.a:

Take the elements from the user and sort them in descending form and do the following:

a. Using binary search, find the element and the location in the array, where the element is asked from user.

Code:

```
#include<stdio.h>
// Bubble sorting in descending order
void sort (int arr[], int n)
{
    int i, j, swap;
    for(i=0; i<n-1; i++)
    {
        for (j=0; j<n-i-1; j++)
        {   if(arr[j] < arr[j+1])
            {   swap = arr[j];
                arr[j]= arr[j+1];
                arr[j+1]= swap;
            }
        }
    }
}
int main()
{   int i, j, low, high, middle, n, search, swap;
    printf("Enter number of elements:");
    scanf("%d",&n);
    int array[n];
```

```
    Printf("Enter the elements :\n");
    for (i=0; i<n; i++)
        scanf("%d", &array[i]);
    Printf("The array in descending order is:");
    sort (array, n);
    for (i=0; i<n; i++)
    {
        printf("%d\t", array[i]);
    }
    printf("\n Enter value to be searched:");
    scanf("%d", &search);
    low = 0;
    high = n-1;
// Searching begins
while (low<=high)
{
        middle = (low+high)/2;
        if (array[middle] < search)
            low = middle+1;
        else if(array[middle] == search)
        {
            printf("%d found at location %d.", search, middle+1);
            break;
        }
        else
            high = middle-1;
}
if(low > high)
    printf("Not found!");
return 0;
}
```

Output:

```
Enter number of elements :5
Enter the elements:
2
3
1
5
4
The array in descending order is : 5  4  3  2  1
Enter value to be searched: 1
1 found at location 5.
```

Program 1.b :

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

Code:

```
#include<stdio.h>
// Bubble sorting in descending order
void sort(int arr[], int n)
{   int i, j, swap;
    for(i=0; i<=n-1; i++)
    {
      for (j=0; j<n-i-1; j++)
      {  if(arr[j] < arr[j+1])
         {   swap = arr[j];
             arr[j] = arr[j+1];
             arr[j+1] = swap;
         }
      }
    }
}
```

```
Void calculation(int a, int b, int arr[])
{
    int sum, product;
    sum = arr[a-1] + arr[b-1];
    product = arr[a-1] * arr[b-1];
    Printf("Sum: %d\n Product: %d", sum, product);
}
int main()
{
    int i, k, l, n;
    Printf("Enter number of elements:");
    scanf("%d", &n);
    int array[n];
    printf("Enter the elements:\n");
    for(i=0; i<n; i++)
        scanf("%d", &array[i]);
    Printf("The array in descending order is:");
    sort(array, n);
    for(i=0; i<n; i++)
        printf("%d\t", array[i]);
    printf("\n Enter two positions:\n");
    scanf("%d\n%d", &k, &l);
    calculation(k, l, array);
    return 0;
}
```

Output:

```
Enter number of elements : 4
Enter the elements:
2
3
1
4
The array in descending order is: 4 3 2 1
Enter two positions:
1
2
Sum: 7
Product: 12
```

Program 2:

Sort the array using merge sort where elements are taken from the user and find the product of k^{th} elements from first and last, where k is taken from the user.

Code:

```
// C program for merge sort
#include <stdio.h>
#include <stdlib.h>

// Merges two subarrays of arr[]
void merge (int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m-l+1;
    int n2 = r-m;

// Create temp arrays
  int L[n1], R[n2];

// Copy data to temp arrays L[] and R[]
  for (i=0; i<n1; i++)
      L[i] = arr[l+i];
  for (j=0; j<n2; j++)
     R[j] = arr[m+1+j];

// Merge the temp arrays back into arr[l....r]
  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++;
    }
    // Copy the remaining elements of L[]
    while (i<n1)
    {   arr[k] = L[i];
        i++;
        k++;
    }
    // Copy the remaining elements of R[]
    while (j<n2)
    {   arr[k] = R[j];
        j++;
        k++;
    }
}
// l is for left index and r is right index of the subarray of arr
void mergeSort(int arr[], int l, int r)
{   if(l<r)
    {   int m = l+(r-l)/2;
        //Sort first and second halves
        mergeSort(arr, l, m);
        mergeSort(arr, m+1, r);
        merge(arr, l, m, r);
    }
}
void printArray( int A[], int size)
{   int i;
    for (i=0; i<size; i++)
        printf("%d ", A[i]);
    printf("\n");
}
```

```
int multiply (int x, int a[])
{
    int i, p=1;
    if (x==0)
        return a[0];
    else
    { for (i=0; i<x; i++)
        p* = a[i];
      return p;
    }
}

int main ()
{
    int n, i, k;
    printf ("Enter the size of the array : ");
    scanf ("%d", &n);
    int arr[n];
    printf (" Enter the elements:\n");
    for (i=0; i<n; i++)
        scanf("%d", &arr[i]);
    mergeSort (arr, 0, n-1);
    printf ("\n Sorted array is \n");
    printArray (arr, n);
    printf ("Enter k:");
    scanf ("%d", &k);
    printf ("The product is %d", multiply (k, arr));
    return 0;
}
```

Output:

```
Enter the size of the array : 6
Enter the elements:
3
4
2
6
1
5
```

Sorted array is
1 2 3 4 5 6
Enter K: 4
The product is : 24

Program 3:

Discuss insertion sort and selection sort with examples.

Answer:

INSERTION SORT:

One element from the array is selected and is compared to one side of the array and inserted to the proper position while shifting the rest of the elements accordingly.

Example:

Let the unsorted array be:

14	33	27	10	35	19	42	44

Insertion sort compares the first two elements.

14	33	27	10	35	19	42	44

It finds that both 14 and 33 are already in ascending order. For now, 14 is in sorted sub-list.

Insertion sort moves ahead and compares 33 with 27. And, finds that 33 is not in the correct position.

It swaps 33 with 27. It also checks with all the elements of the sorted sub-list.

14	27	33	10	35	19	42	44

By now we have 14 and 27 in the sorted sub-list. Next, it compares 33 with 10. As they are not in order, we swap them.

14	27	10	33	35	19	42	44

However, swapping makes 27 and 10 unsorted.

14	27	10	33	35	19	42	44

Hence, we swap them too.

14	10	27	33	35	19	42	44

Again we find 14 and 10 in an unsorted order.
We swap them again. By the end of the third iteration, we have a sorted sub-list of 4 items.

10	14	27	33	35	19	42	44

This process goes on until all the unsorted values are covered in a sorted sub-list.

10	14	19	27	33	35	42	44

SELECTION SORT:

It is basically a selection of an element position from the start with the other rest of the elements. Elements are compared and exchanged depending on the condition and then selection position is shifted to the next position till it reaches to the end.

EXAMPLE:

Consider the following unsorted array:

14	33	27	10	35	19	42	44

For the first position in the sorted list, the whole list is scanned sequentially. The first position where 14 is stored presently, we search the whole list and find that 10 is the lowest value.
So, we replace 14 with 10.

10	33	27	14	35	19	42	44

For the second position, where 33 is residing, we start scanning the rest of the list.
We find that 14 is the second lowest value in the list and it should appear at the second place. We swap these values.
After two iterations, two least values are positioned at the beginning in a sorted manner.

10	14	27	33	35	19	42	44

The same process is applied to rest of the items in the array.

10	14	19	27	33	35	42	44

Program 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.

Code :

```
#include<stdio.h>
// Bubble sort in descending order
void sort(int arr[], int n)
{   int i, j, swap;
    for (i=0; i<n-1; i++)
    {   for (j=0; j<n-i-1; j++)
        {   if (arr[j] < arr[j+1])
            {   swap = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = swap;
            }
        }
    }
}
```

```
// Function to print the sorted array in alternate order
void Print(int arr[], int n)
{
    int i;
    Printf("\n Alternate order:\n");
    for(i=0; i<=n; i+=2)
        Printf("%d \t", arr[i]);
}
// Function to print sum of elements in odd positions and
  Product of elements in even positions
Void calculation(int arr[], int k)
{
    int i, sum=0, product=1;
    for(i=0; i<k; i++)
    {   if((i+1)%2==0)
            product*=arr[i];
        else
            sum+=arr[i];
    }
    printf("\n Sum of elements in odd positions: %d\n Product
    of elements in even positions: %d", sum, product);
}
// Function to check if any element is divisible by K
void divisible(int arr[], int x, int k)
{   int i;
    printf("\n The elements divisible by %d are:\n", k);
    for(i=0; i<x; i++)
    {   if(arr[i]%k==0)
            printf("%d\t", arr[i]);
    }
}
int main()
{   int i,n,m;
```

```
    Printf ("Enter number of elements: ");
    scanf ("%d", &n);
    int array[n];
    Printf ("Enter the elements :\n");
    for (i=0; i<n; i++)
        scanf ("%d", &array[i]);
    Printf ("\n The array in descending order is :\n");
    sort (array, n);
    for (i=0; i<n; i++)
        Printf ("%d \t", array[i]);
    Print (array, n);
    calculation (array, n);
    Printf ("Enter the value of m : ");
    scanf ("%d", &m);
    divisible (array, n, m);
    return 0;
}
```

Output:

```
Enter number of elements: 5
Enter the elements:
2
3
1
5
4
The array in descending order is:
 5   4   3   2   1
 Alternate order:
 5   3   1
Sum of elements in odd positions: 9
Product of elements in even positions: 8
Enter the value of m: 2

The elements divisible by 2 are:
 4   2
```

Program 5:
Write a recursive program to implement binary search.
Code:

```
#include<stdio.h>
void binary_search(int[], int, int, int);
void bubble_sort (int[], int);
int main()
{
    int key, size, i;
    int list[25];
    Printf("Enter size of a list(n<25):");
    Scanf("%d", &size);
    Printf("Enter elements\n");
    for(i=0; i<size; i++)
        scanf("%d", &list[i]);
    bubble_sort(list, size);
    printf("\n");
    printf("Enter key to search\n");
    scanf("%d", &key);
    binary_search(list, 0, size, key);
}
void bubble_sort(int list[], int size)
{   int temp, i, j;
    for(i=0; i<size; i++)
    {  for(j=i; j<size; j++)
       {    if(list[i] > list[j])
            {   temp = list[i];
                list[i] = list[j];
                list[j] = temp;
            }
       }
    }
}
```

```
Void binary_search (int list[], int first, int last, int key)
{
    int mid;
    if (first > last)
    {
        printf ("Number not found!\n");
        return;
    }
    mid = (first + last) /2;
    if (list[mid] == key)
    {
        printf ("Number found!\n");
    }
    else if (list[mid] > key)
        binary_search (list, last, mid-1, key);
    else if (list[mid] < key)
        binary_search (list, mid+1, last, key);
}
```

Output:

```
Enter size of a list : 5
Enter elements:
2
3
1
5
4
Enter key to search:
5
Number found!
```