DSA LAB 2

```
//Q1.WAP to reverse the contents of a array of n elements.
#include <stdio.h>
int main()
{
    int n,i,t;
    printf("Enter number of element in array:");
    scanf("%d",&n);
    int a[n];

    printf("\n\nThe integers entered are: \n");

    for(i=0; i<n; i++)
    scanf("%d",&a[i]);

    for(int i = 0; i<n/2; i++){
        t = a[i];
        a[i] = a[n-i-1];
        a[n-i-1] = t;
}
   printf("\nReversed array:");
   for(int i = 0; i < n; i++)

        printf("%d  ", a[i]);

}
```

OUTPUT

```
Enter number of element in array:5
The integers entered are:
1
2
3
4
5
Reversed array:5  4  3  2  1
```

```
//Q.2 WAP to search an element in array of n numbers.

#include <stdio.h>
int search(int *a, int n, int key)
{
int i;
for (i = 0; i < n; i++)
{
if (a[i] == key)
{
return 1;
}
}
return 0;
}
int main()
{
int a[10000], i, n, key;
printf("Enter size of the array : ");
scanf("%d", &n);
printf("Enter elements in array : ");
for (i = 0; i < n; i++)
{
scanf("%d", &a[i]);
}
printf("Enter the key : ");
scanf("%d", &key);
if (search(a, n, key))
printf("element found ");
else
printf("element not found ");
}
```

OUTPUT

```
Enter size of the array : 4
Enter elements in array : 2
3
45
34
Enter the key : 34
element found
```

```
//Q3. WAP to display the array elements in descending order.
#include <stdio.h>
void main ()
{
  int i, j, t, n;
  printf("enter number of elements in an array\n");
  scanf("%d", &n);
  int a[n];
  printf("Enter the elements\n");

  for (i = 0; i < n; ++i)
    scanf("%d", &a[i]);

  for (i = 0; i < n; ++i)
  {
    for (j = i + 1; j < n; ++j)
    {
      if (a[i] < a[j])
      {
        t = a[i];
        a[i] = a[j];
        a[j] = t;
      }
    }
  }
  printf("The numbers in descending order is:\n");
  for (i = 0; i < n; ++i)
  {
    printf("%d\n", a[i]);
  }
}
```

OUTPUT

```
enter number of elements in an array
3
Enter the elements
4
67
93
The numbers in descending order is:
93     67     4
```

```
/*Q4.Given an unsorted array of size n, WAP to find and display the
number of elements between two elements a and b (both inclusive). E.g.
Input : arr = [1, 2,2, 7, 5, 4], a=2 and b=5, Output : 4 and the numbers are:
2, 2, 7, 5.*/
#include <stdio.h>

int main()
{
    int n, i, a, b, c = 0;

    printf("Enter size of array: ");
    scanf("%d", &n);

    printf("Enter elements of array: ");

    int arr[n];

    for (i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }

    printf("\nEnter lower limit element & upper limit element respectively:
");
    scanf("%d %d", &a, &b);
    int arj[10];
    for (i = 0; i < n; i++)
    {
        if (arr[i] == a || arr[i] == b)
        {
            c++;
        }

        if (arr[i] > a && arr[i] < b)
        {
            c++;
        }
    }
    printf("Number of elements in between two elements (Both Inclusive)
= %d", c);
```

```
   print("")

   return 0;
}
```

OUTPUT

```
Enter size of array: 6
Enter elements of array: 1
2
2
7
5
4
Enter lower limit element & upper limit element respectively: 2
5
Number of elements in between two elements (Both Inclusive) = 4
```

```
/*Q5.Given a array, WAP to print the next greater element (NGE) for
every element.
The next greater element for an element x is the first greater element on
the right
side of x in array. Elements for which no greater element exist, consider
next
greater element as -1. E.g. For the input array [2, 5, 3, 9, 7], the next
greater
elements for each elements are as follows.*/
#include<stdio.h>
void printNGE(int arr[], int n)
{
        int next, i, j;
        for (i=0; i<n; i++)
        {
                next = -1;
                for (j = i+1; j<n; j++)
                {
                        if (arr[i] < arr[j])
                        {
                                next = arr[j];
                                break;
                        }
                }
                printf("%d\n", next);
        }
}

int main()
{
        int arr[]= {2, 5, 3, 9, 7};
        int n = sizeof(arr)/sizeof(arr[0]);
        printNGE(arr, n);
        return 0;
}
```

OUTPUT

```
5
9
9
-1
-1
```

```
/*6. Given an unsorted array arr and two numbers x and y, find the
minimum
distance between x and y in arr. The array might also contain duplicates.
You may assume that both x and y are different and present in arr. Input:
arr[] ={3, 5, 4, 2, 6, 5, 6, 6, 5, 4, 8, 3}, x = 3, y = 6 Output: Minimum
distance
between 3 and 6 is 4.*/
#include <stdio.h>
void minDistance(int *Arr, int n, int x, int y)
{
int distance[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, a = 0, s = 0;
;
for (int i = 0; i <= n; i++)
{
if ((Arr[i] == x || Arr[i] == y) && s == 0)
{
if (Arr[i] == x)
{
s = 1;
// printf("Start by %d",Arr[i]);
}
else if (Arr[i] == y)
{
s = 2;
// printf("Start by %d\n",Arr[i]);
}
}
else if (s == 1 && Arr[i] == y)
{
s = 0;
a++;
// printf("end by %d\n",Arr[i]);
}
else if (s == 2 && Arr[i] == x)
{
s = 0;
a++;
// printf("end by %d\n",Arr[i]);
}
if (s != 0)
distance[a]++;
}
int min = distance[0];
for (int i = 0; i < n; i++)
{
if (distance[i] <= min && distance[i] != 0)
min = distance[i];
}
printf("Min Distance between %d and %d is %d\n", x, y, min);
}
int main()
{
int Arr[50] = {3, 5, 4, 2, 6, 5, 6, 6, 5, 4, 8, 3}, n = 12, a, b;
// limit from a to b.
a = 3;
b = 6;
minDistance(Arr, n, a, b);
return 0;
}
```

Output

```
Min Distance between 3 and 6 is 4
```

```
//7. WAP to arrange the elements of a array such that all even numbers
are
//followed by all odd numbers.
#include <stdio.h>
void swap(int *a, int *b);
void segregateEvenOdd(int arr[], int size)
{
int left = 0, right = size - 1;
while (left < right)
{
while (arr[left] % 2 == 0 && left < right)
left++;
while (arr[right] % 2 == 1 && left < right)
right--;
if (left < right)
{
swap(&arr[left], &arr[right]);
left++;
right--;
}
}
}
void swap(int *a, int *b)
{
int temp = *a;
*a = *b;
*b = temp;
}
int main()
{
int arr[] = {21,34,56,79,89,45,50};
int arr_size = sizeof(arr) / sizeof(arr[0]);
int i = 0;
segregateEvenOdd(arr, arr_size);
printf("Array after segregation ");
for (i = 0; i < arr_size; i++)
printf("%d ", arr[i]);
return 0;
}
```

OUTPUT

```
Array after segregation 50 34 56 79 89 45 21
```

```
//8. let a be nXn square matrix. WAP by using appropriate user defined
//functions for the following: a) find the number of nonzero elements in
//A b) find the sum of the elements above the leading diagonal. c)
///Display the elements below the mirror diagonal. d) find the product
of
//the diagonal elements.
#include <stdio.h>
void NonZeroElements(int (*Arr)[5], int n)
{
int a = 0;
for (int i = 0; i < n; i++)
{
for (int j = 0; j < n; j++)
{
if (*(*(Arr + i) + j) != 0)
a++;
}
}
printf("Number of non-zero elements in array: %d\n", a);
}
void sumOfElementsAboveLeadingDiagonal(int (*Arr)[5], int n)
{
int a;
for (int i = 0; i < n; i++)
{
for (int j = 0; j < n; j++)
{
if (j > i)
a += *(*(Arr + i) + j);
}
}
printf("Sum Of Elements Above Leading Diagonal: %d\n", a);
```

```
}
void ElementsBelowMinorDiagonal(int (*Arr)[5], int n)
{
printf("Elements below the minor diagonal: \n");
for (int i = 0; i < n; i++)
{
for (int j = 0; j < n; j++)
{
if (j >= n - i)
printf(" %d", *(*(Arr + i) + j));
else
printf(" ");
}
printf("\n");
}
}
void ProductOfdiagonalElements(int (*Arr)[5], int n)
{
int a = 1;
for (int i = 0; i < n; i++)
{
for (int j = 0; j < n; j++)
{
if (i == j)
a *= *(*(Arr + i) + j);
}
}
printf("Product of leading diagnal Elements: %d\n", a);
}
int main()
{
int mat[5][5] = {
{00, 01, 02, 03, 04},
{10, 11, 12, 13, 14},
{20, 21, 22, 23, 24},
{30, 31, 32, 33, 34},
{40, 41, 42, 43, 44},
};
int n = 5;
NonZeroElements(mat, n);
```

```
sumOfElementsAboveLeadingDiagonal(mat, n);
ElementsBelowMinorDiagonal(mat, n);
ProductOfdiagonalElements(mat, n);
}
```

OUTPUT

```
Number of non-zero elements in array: 24
Sum Of Elements Above Leading Diagonal: 154
Elements below the minor diagonal:

    14
   23 24
  32 33 34
 41 42 43 44
Product of leading diagnal Elements: 0
```

RISHIKESH
2105734
CSE32