ALWIN TOMY

2347207

Problems using Structures

1)Given an integer N and an integer D, rotate the binary representation of the integer N by D digits to the left as well as right and return the results in their decimal representation after each of the rotation.
Note: Integer N is stored using 16 bits. i.e. 12 will be stored as 0000000000001100.

Input:

N = 28, D = 2

Output:

112

7

Explanation:

28 in Binary is: 0000000000011100

Rotating left by 2 positions, it becomes 0000000001110000 = 112 (in decimal).

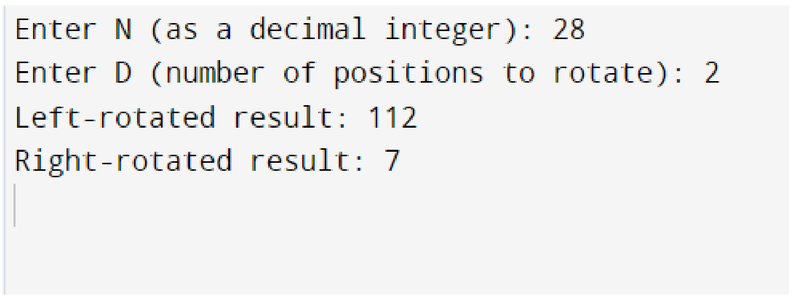
Rotating right by 2 positions, it becomes 0000000000000111 = 7 (in decimal).

PROGRAM :

```
#include <stdio.h>

unsigned short leftRotate(unsigned short n, int d) {
   return (n << d) | (n >> (16 - d));
}

unsigned short rightRotate(unsigned short n, int d) {
   return (n >> d) | (n << (16 - d));
}

int main() {
   unsigned short N;
```

```
    int D;

    printf("Enter N (as a decimal integer): ");
    scanf("%hu", &N);
    printf("Enter D (number of positions to rotate): ");
    scanf("%d", &D);

    if (D < 0) {
        D = 16 - (-D % 16);
    } else {
        D = D % 16;
    }

    unsigned short leftRotated = leftRotate(N, D);
    printf("Left-rotated result: %hu\n", leftRotated);

    unsigned short rightRotated = rightRotate(N, D);
    printf("Right-rotated result: %hu\n", rightRotated);

    return 0;
}
```

Output:



```
Enter N (as a decimal integer): 28
Enter D (number of positions to rotate): 2
Left-rotated result: 112
Right-rotated result: 7
```

2) Given an array arr[] of positive integers of size N. Reverse every sub-array group of size K.

Note: If at any instance, there are no more subarrays of size greater than or equal to K, then reverse the last subarray (irrespective of its size). You shouldn't return any array, modify the given array in-place.

Input:

N = 5, K = 3

arr[] = {1,2,3,4,5}

Output: 3 2 1 5 4

Explanation: First group consists of elements

1, 2, 3. Second group consists of 4,5.

input:

N = 4, K = 3

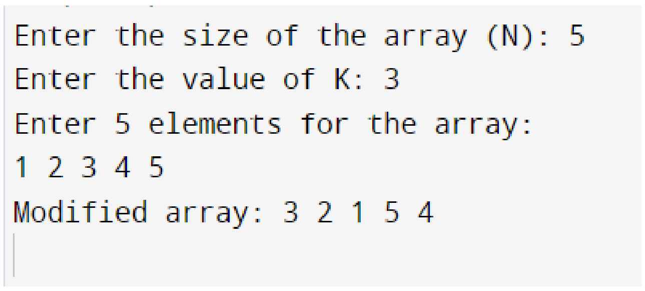
arr[] = {5,6,8,9}

Output: 8 6 5 9

PROGRAM :

```
#include <stdio.h>

void reverseArray(int arr[], int start, int end) {
    while (start < end) {
        // Swap elements at start and end positions
        int temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;
        start++;
        end--;
    }
}

void reverseSubArrays(int arr[], int N, int K) {
    int i;
    for (i = 0; i < N; i += K) {
        int end = i + K - 1;

        if (end >= N) {
            end = N - 1;
        }

        reverseArray(arr, i, end);
    }
}

int main() {
    int N, K;

    printf("Enter the size of the array (N): ");
    scanf("%d", &N);
    printf("Enter the value of K: ");
    scanf("%d", &K);

    int arr[N];

    printf("Enter %d elements for the array:\n", N);
    for (int i = 0; i < N; i++) {
        scanf("%d", &arr[i]);
    }

    reverseSubArrays(arr, N, K);

    printf("Modified array: ");
    for (int i = 0; i < N; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

Output:



```
Enter the size of the array (N): 5
Enter the value of K: 3
Enter 5 elements for the array:
1 2 3 4 5
Modified array: 3 2 1 5 4
```

3) Given two arrays A and B of equal size N, the task is to find if given arrays are equal or not. Two arrays are said to be equal if both of them contain same set of elements, arrangements (or permutation) of elements may be different though.

Note : If there are repetitions, then counts of repeated elements must also be same for two array to be equal.

Input:

N = 5

A[] = {1,2,5,4,0}

B[] = {2,4,5,0,1}

Output: 1

Explanation: Both the array can be

rearranged to {0,1,2,4,5}

Example 2:

Input:

N = 3

A[] = {1,2,5}

B[] = {2,4,15}

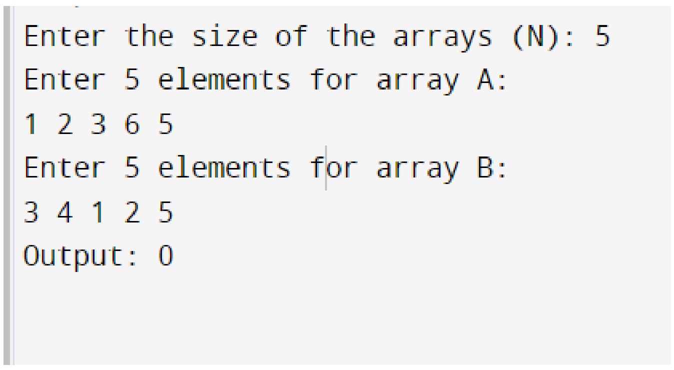
Output: 0

Explanation: A[] and B[] have only one common value

PROGRAM :

```
#include <stdio.h>
#include <stdlib.h>
int areArraysEqual(int A[], int B[], int N) {
   int countA[101] = {0};
   int countB[101] = {0};

   for (int i = 0; i < N; i++) {
      countA[A[i]]++;
   }

   for (int i = 0; i < N; i++) {
      countB[B[i]]++;
   }
   for (int i = 0; i <= 100; i++) {
      if (countA[i] != countB[i]) {
          return 0;
      }
   }
   return 1;
}
int main() {
   int N;

   printf("Enter the size of the arrays (N): ");
   scanf("%d", &N);

   int A[N], B[N];

   printf("Enter %d elements for array A:\n", N);
   for (int i = 0; i < N; i++) {
      scanf("%d", &A[i]);
   }

   printf("Enter %d elements for array B:\n", N);
   for (int i = 0; i < N; i++) {
      scanf("%d", &B[i]);
   }

   int result = areArraysEqual(A, B, N);

   printf("Output: %d\n", result);

   return 0;
}
```

Output:



```
Enter the size of the arrays (N): 5
Enter 5 elements for array A:
1 2 3 6 5
Enter 5 elements for array B:
3 4 1 2 5
Output: 0
```

4) Given an integer array Arr of size N. For each element in the array, check whether the right adjacent element (on the next immediate position) of the array is smaller. If next element is smaller, update the current index to that element. If not, then -1.

Input:

N = 5

Arr[] = {4, 2, 1, 5, 3}

Output:

2 1 -1 3 -1

Explanation: Array elements are 4, 2, 1, 5

Next to 4 is 2 which is smaller, so we print 2. Next of 2 is 1 which is smaller, so we print 1. Next of 1 is 5 which is greater, so we print -1. Next of 5 is 3 which is smaller, so we print 3. Note that for last element, output is always going to be -1 because there is no element on right.

Input:

N = 6

Arr[] = {5, 6, 2, 3, 1, 7}

Output:

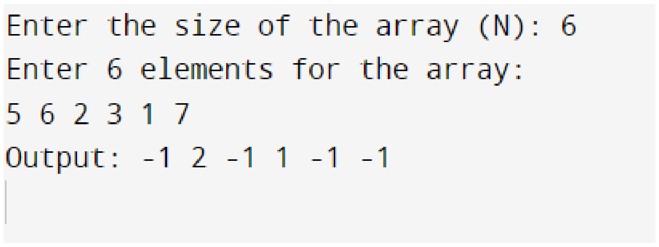
-1 2 -1 1 -1 -1

Explanation: Next to 5 is 6 which is greater, so we print -1.Next of 6 is 2 which is smaller, so we print 2. Next of 2 is 3 which is greater, so we print -1. Next of 3 is 1 which is smaller, so we print 1. Next of 1 is 7 which is greater, so we print -1. Note that for last element, output is always going to be -1 because there is no element on right.

PROGRAM :

```
#include <stdio.h>
int main() {
   int N;

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

   int Arr[N];

   printf("Enter %d elements for the array:\n", N);
   for (int i = 0; i < N; i++) {
       scanf("%d", &Arr[i]);
   }

   int result[N];
   for (int i = 0; i < N; i++) {
       result[i] = -1;
   }

   for (int i = 0; i < N - 1; i++) {
       if (Arr[i] > Arr[i + 1]) {
           result[i] = Arr[i + 1];
       }
   }

   printf("Output: ");
   for (int i = 0; i < N; i++) {
       printf("%d ", result[i]);
   }
   printf("\n");

   return 0;
}
```

Output:



```
Enter the size of the array (N): 6
Enter 6 elements for the array:
5 6 2 3 1 7
Output: -1 2 -1 1 -1 -1
```