1. Given a number ‘n’ and a position ‘p’, write an algorithm and define a function using ‘C’ program to check if the ‘p-th’ digit from right of ‘n’ is odd or even. For example, if ‘n’ is 3145782 and p is 4 then you have to check if 5 is odd or even. Since it is even print ‘Odd’. Make your code accept numbers of larger size.

Input Format:

The first line contains the number, n

The second line contains the position, p

Output Format:

Print either “Odd” or “Even”

#include <stdio.h>

int main()

{

    long n;

    int i, p;

    scanf("%ld %d", &n, &p);

    for (i = 1; i < p; i++)

        n /= 10;

    if ((n % 10) % 2)

        printf("Odd");

    else

        printf("Even");

    return 0;

}

INPUT

3145782

4

OUTPUT

Odd

2. Given a number ‘n’ and a position ‘p’, write an algorithm and define a function using ‘C’ program to check if the ‘p-th’ digit from left of ‘n’ is odd or even. For example, if ‘n’ is 3145782 and p is 4 then you have to check if 5 is odd or even. Since it is odd print ‘Odd’. Make your code accept numbers of larger size.

Input Format:

The first line contains the number, n

The second line contains the position, p

Output Format:

Print either “Odd” or “Even”

#include <stdio.h>

int main()

{

    long n, t;

    int i, p, l = 0;

    scanf("%ld %d", &n, &p);

    t = n;

    while (t > 1)

    {

        t /= 10;

        l++;

    }

    t = 10;

    for (i = 0; i < p; i++)

            t \*= 10;

    n /= t;

    if ((n % 10) % 2)

        printf("Odd");

    else

        printf("Even");

    return 0;

}

INPUT

3145782

4

OUTPUT

Odd

3. Given an array of integers 'arr' as an input. Write a program that will print all the top elements in the array. An element is said to be top if it is greater than all the elements to its right side. And the rightmost element is always a top element.

Input format:

Read n #no of elements to be stored in array

Read s # elements to be stored in array

Output format:

Print the digits which are top elements.

For example:

Input: arr = [6, 7, 4, 3, 5, 2]

Output: top elements are 7, 5 and 2.

#include <stdio.h>

#include <stdlib.h>

int main()

{

    int i, j, n;

    scanf("%d", &n);

    int \*a = (int \*)malloc(sizeof(int)\*n);

    for (i = 0; i < n; i++)

        scanf("%d", &a[i]);

    for (i = 0; i < n; i++)

    {

        int chk = 1;

        for (j = i + 1; j < n; j++)

            if (a[i] <= a[j])

            {

                chk = 0;

                break;

            }

        if (chk)

            printf("%d ", a[i]);

    }

    return 0;

}

INPUT

6

6 7 4 3 5 2

OUTPUT

7 5 2

4. Given a square matrix of size N×N, devise an algorithm and write the Python code to calculate the absolute difference of the sums across the two main diagonals. For example, given a 3 X 3 matrix as below:

11 2 4

4 5 6

10 8 -12

Sum across the first diagonal = 11+5-12= 4

Sum across the second diagonal = 4+5+10 = 19

Absolute Difference: |4-19| =15

Input format:

Enter the value of n (nxn matrix)

Enter the matrix element

Output format:

Sum across the first diagonal

Sum across the second diagonal

Absolute Difference

#include <stdio.h>

#include <stdlib.h>

int main()

{

    int main = 0, off = 0, i, n;

    printf("Enter order of matrix: ");

    scanf("%d", &n);

    int \*a = (int \*)malloc(sizeof(int)\*n\*n);

    printf("Entire matrix elements (row-wise) below:\n");

    for (i = 0; i < n\*n; i++)

        scanf("%d", &a[i]);

    for (i = 0; i < n\*n; i += (n + 1))

        main += a[i];

    for (i = n - 1; i < n\*n - 1; i += (n - 1))

        off += a[i];

    printf("Sum across the first diagonal: %d\nSum across the second diagonal: %d\nAbsolute difference: %d", main, off, abs(main - off));

    return 0;

}

INPUT

Enter order of matrix: 3

Entire matrix elements (row-wise) below:

11 2 4

4 5 6

10 8 -12

OUTPUT

Sum across the first diagonal: 4

Sum across the second diagonal: 19

Absolute difference: 15