1.    Write an expression that checks whether an integer is **odd or even**.

using System;

namespace detyra1

{

class Program

{

static void Main(string[] args)

{

int i;

Console.Write("Give a Number : ");

i = int.Parse(Console.ReadLine());

if (i % 2 == 0)

{

Console.Write("Entered Number is an Even Number");

Console.Read();

}

else

{

Console.Write("Entered Number is an Odd Number");

Console.Read();

}

}

}

}

2.    Write a Boolean expression that checks whether a given integer is **divisible by both 5 and 7**, without a remainder.

using System;

class DivideBySevenAndFive

{

static void Main()

{

Console.WriteLine("Please enter your number:");

double n = double.Parse(Console.ReadLine());

bool result = (n % 5 == 0) && (n % 7 == 0);

Console.WriteLine("The number {0} can be divided by 5 and 7? ANSWER: {1}", n, result);

}

}

3.    Write an expression that checks for a given integer if its **third digit** (right to left) is 7.

using System;

class ThirdDigit

{

static void Main()

{

Console.Write("Enter a number: ");

int number = int.Parse(Console.ReadLine());

int thirdDigit = (number / 100) % 10;

if (thirdDigit == 7)

{

Console.WriteLine("The third digits IS seven!");

}

else

{

Console.WriteLine("The third digit IS NOT seven.");

}

}

}

4.    Write an expression that checks whether the **third bit** in a given integer is 1 or 0.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace detyra4

{

class Program

{

static void Main()

{

bool is3Digit1 = false;

Console.Write("Enter the number:");

int number;

bool isNumber = int.TryParse(Console.ReadLine(), out number);

if (isNumber)

{

if ((number & 8) == 8)

{

is3Digit1 = true;

}

Console.WriteLine("Is the third digit equal to 1?:{0}", is3Digit1);

}

else

{

Console.WriteLine("Not a valid entry!");

}

}

}

}

5.    Write an expression that calculates the **area of a trapezoid** by given sides **a**, **b** and height **h**.

using System;

class Trapezoid

{

static void Main()

{

Console.WriteLine("Base a");

double a = double.Parse(Console.ReadLine());

Console.WriteLine("Base b");

double b = double.Parse(Console.ReadLine());

Console.WriteLine("Height");

double h = double.Parse(Console.ReadLine());

double area = ((a + b) / 2) \* h;

Console.WriteLine("The area of your trapezoid is: {0}", area);

}

}

6.    Write a program that prints on the console the **perimeter and the area of a rectangle**by given side and height entered by the user.

using System;

namespace Area\_Perimeter

{

class Program

{

static void Main(string[] args)

{

int l, w;

Console.Write("Enter the length of Rectangle: ");

l = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter the width of Rectangle: ");

w = Convert.ToInt32(Console.ReadLine());

int area = l \* w;

int peri = (2 \* l) + (2 \* w);

Console.WriteLine("Area of Rectangle is = {0}" , area);

Console.WriteLine("Perimeter of Rectangle is = {0}", peri);

Console.ReadKey();

}

}

}

7.    The gravitational field of the Moon is approximately 17% of that on the Earth. Write a program that calculates the **weight of a man on the moon** by a given weight on the Earth.

using System;

class GravitationOnTheMoon

{

static void Main()

{

Console.WriteLine("Please, enter the weight of the person: ");

double weight = double.Parse(Console.ReadLine());

Console.WriteLine("The weight on the moon is: {0}", weight \* 17 / 100 );

}

}

8.    Write an expression that checks for a given point {x, y} if it is **within the circle**K[{0, 0}, R=5]. Explanation: the point {0, 0} is the center of the circle and 5 is the radius.

using System.Text;

namespace \_6.IsThePointWithinTheCircle

{

class Program

{

static void Main()

{

bool check = false;

double X, Y;

Console.Write("Enter X:");

bool Xcheck = double.TryParse(Console.ReadLine(), out X);

Console.Write("Enter Y:");

bool Ycheck = double.TryParse(Console.ReadLine(), out Y);

if (Xcheck && Ycheck)

{

if ((X \* X + Y \* Y) <= 25)

{

check = true;

}

Console.WriteLine("Is the point ({0},{1}) within the circle?:{2}", X, Y, check);

}

else

{

Console.WriteLine("Not a valid entry!");

}

}

}

}

9.    Write an expression that checks for given point {x, y} if it is **within the circle** K[{0, 0}, R=5] and **out of the rectangle** [{-1, 1}, {5, 5}]. Clarification: for the rectangle the lower left and the upper right corners are given.

using System;

public class Program

{

public static void Main()

{

int x = int.Parse(Console.ReadLine());

int y = int.Parse(Console.ReadLine());

int R = 5;

if ((x \* x) + (y \* y) <= R \* R && (x < -1 || y < 1))

{

Console.WriteLine("Point {{{0} , {1}}} is inside the circle with radius {1} and outside of rectangle.", x, y, R);

}

else

{

Console.WriteLine("Point {{{0} , {1}}} is outside the circle with radius {1}.", x, y, R);

}

}

}

10.   Write a program that takes as input a **four-digit number** in format **abcd** (e.g. 2011) and performs the following actions:

-     Calculates the sum of the digits (in our example 2+0+1+1 = 4).

-     Prints on the console the number in reversed order: **dcba** (in our example 1102).

-     Puts the last digit in the first position: **dabc** (in our example 1201).

-     Exchanges the second and the third digits: **acbd** (in our example 2101).

using System;

class Program

{

static void Main()

{

int n = int.Parse(Console.ReadLine());

int firstNumber = (n / 1000) % 10;

int secondNumber = (n / 100) % 10;

int thirdNumber = (n / 10) % 10;

int fourthNumber = (n % 10);

Console.WriteLine("The sum of the digits is: {0}", firstNumber + secondNumber + thirdNumber + fourthNumber);

Console.WriteLine("Reversed order: {0}{1}{2}{3}", fourthNumber, thirdNumber, secondNumber, firstNumber);

Console.WriteLine("Last digit upfront: {0}{1}{2}{3}", fourthNumber, firstNumber, secondNumber, thirdNumber);

Console.WriteLine("Exchanges the second and the third digits: {0}{1}{2}{3}", firstNumber, thirdNumber, secondNumber, fourthNumber);

}

}

11.   We are given number **n** and position **p**. Write a sequence of operations that prints the value of **the bit on the position** **p** in the number (0 or 1). Example: **n**=35, **p**=5 -> 1. Another example: n=35, **p**=6 -> 0.

using System;

namespace \_11.FindInAGivenValueGivenBitNumber

{

class Program

{

static void Main()

{

byte value = 0;

int i;

byte p;

Console.Write("Enter the integer number i:");

bool isiInt = int.TryParse(Console.ReadLine(), out i);

Console.Write("Enter the position of the bit p:");

bool isbByte = byte.TryParse(Console.ReadLine(), out p);

if (isiInt && isbByte && p < 32)

{

int mask = 1 << p;

if ((i & mask) == mask)

{

value = 1;

}

Console.WriteLine("value={0}", value);

}

else

{

Console.WriteLine("Not a valid entry!");

}

}

}

}

12.   Write a Boolean expression that checks if the bit on position **p** in the integer **v** has the value 1. Example v=5, **p**=1 -> **false**.

using System;

namespace \_10.CheckIfaBitInIntIs0or1

{

class Program

{

static void Main()

{

bool isDigit1 = false;

byte p;

int v;

Console.Write("Enter the position of the bit p:");

bool ispByte = byte.TryParse(Console.ReadLine(), out p);

Console.Write("Enter the integer number v:");

bool isvInt = int.TryParse(Console.ReadLine(), out v);

if (ispByte && isvInt && p < 32)

{

int mask = 1 << p;

if ((v & mask) == mask)

{

isDigit1 = true;

}

Console.WriteLine("Is bit{0} of intiger {1} equal to 1?:{2}", p, v, isDigit1);

}

else

{

Console.WriteLine("Not a valid entry!");

}

}

}

}

13.   We are given the number **n**, the value **v** (**v** = 0 or 1) and the position **p**. write a sequence of operations that changes the value of **n**, so the bit on the position **p** has the value of **v**. Example: n=35, p=5, v=0 -> n=3. Another example: n=35, p=2, v=1 -> n=39.

using System;

namespace Detyra13

{

class Program

{

static void Main(string[] args)

{

int n = 350;

int v = 0;

int p = 3;

n = (v == 0) ? n = n & (~(1 << p)) : n = n | (1 << p);

Console.WriteLine(n);

}

}

}

14.   Write a program that checks if a given number **n** (1 <**n** < 100) is a **prime number** (i.e. it is divisible without remainder only to itself and 1).

using System;

namespace Detyra14

{

class Program

{

static void Main(string[] args)

{

int number = 72;

bool isPrime = true;

if (number > 2)

for (int i = 2; i <= Math.Ceiling(Math.Sqrt(numbber)); ++i)

{

if (number % i == 0) isPrime = false;

}

Console.WriteLine("{0} is prime?: {1}", number, isPrime);

}

}

}

15.   \* Write a program that **exchanges the values of the bits** on positions 3, 4 and 5 with bits on positions 24, 25 and 26 of a given 32-bit unsigned integer.

using System;

namespace Detyra15

{

class Program

{

static void Main(string[] args)

{

Console.Write("Enter number: ");

int v = Convert.ToInt32(Console.ReadLine());

int mask = 1 << 3;

int bitAt3 = (v & mask) != 0 ? 1 : 0;

mask = 1 << 4;

int bitAt4 = (v & mask) != 0 ? 1 : 0;

mask = 1 << 5;

int bitAt5 = (v & mask) != 0 ? 1 : 0;

mask = 1 << 24;

int bitAt24 = (v & mask) != 0 ? 1 : 0;

mask = 1 << 25;

int bitAt25 = (v & mask) != 0 ? 1 : 0;

mask = 1 << 26;

int bitAt26 = (v & mask) != 0 ? 1 : 0;

v = (bitAt3 == 0) ? v = v & (~(1 << 24)) : v = v | (1 << 24);

v = (bitAt4 == 0) ? v = v & (~(1 << 25)) : v = v | (1 << 25);

v = (bitAt5 == 0) ? v = v & (~(1 << 26)) : v = v | (1 << 26);

v = (bitAt24 == 0) ? v = v & (~(1 << 3)) : v = v | (1 << 3);

v = (bitAt25 == 0) ? v = v & (~(1 << 4)) : v = v | (1 << 4);

v = (bitAt26 == 0) ? v = v & (~(1 << 5)) : v = v | (1 << 5);

Console.WriteLine(v);

}

}

}

16.   \* Write a program that **exchanges bits** {p, p+1, …, p+k-1} with bits {q, q+1, …, q+k-1} of a given 32-bit unsigned integer.

using System;

namespace Detyra16

{

class Program

{

private static uint ModifyNumber(uint number, int p, int q, int k)

{

int[] pBits = new int[k];

int[] qBits = new int[k];

for (int position = p, i = 0; i < pBits.Length; position++, i++)

{

pBits[i] = PthBit(number, position);

}

for (int position = q, i = 0; i < qBits.Length; position++, i++)

{

qBits[i] = PthBit(number, position);

}

for (int position = p, i = 0; i < qBits.Length; position++, i++)

{

number = ModifiedNumber(number, position, qBits[i]);

}

for (int position = q, i = 0; i < pBits.Length; position++, i++)

{

number = ModifiedNumber(number, position, pBits[i]);

}

return number;

}

private static int PthBit(uint number, int position)

{

uint pthBit = (number >> position) & 1;

return (int)pthBit;

}

private static uint ModifiedNumber(uint number, int position, int bitValue)

{

uint actualP = (uint)bitValue << position;

number = number & (~((uint)1 << position));

uint result = number | actualP;

return result;

}

static void Main(string[] args)

{

Console.Write("Enter number: ");

uint number = uint.Parse(Console.ReadLine());

Console.Write("Enter p: ");

int p = int.Parse(Console.ReadLine());

Console.Write("Enter q: ");

int q = int.Parse(Console.ReadLine());

Console.Write("Enter k: ");

int k = int.Parse(Console.ReadLine());

if (p > q)

{

int oldValue = p;

p = q;

q = oldValue;

}

if (p + k >= q)

{

k += p - q - 1;

q += p + k + 1;

}

number = ModifyNumber(number, p, q, k);

Console.WriteLine(number);

}

}

}