**Problem 1:** Write an expression that checks if given integer is **odd or even**. Examples:

|  |  |
| --- | --- |
| **n** | **Odd?** |
| 3 | true |
| 2 | false |

class OddOrEven

{

static void Main()

{

Console.Write("Please write an integer: ");

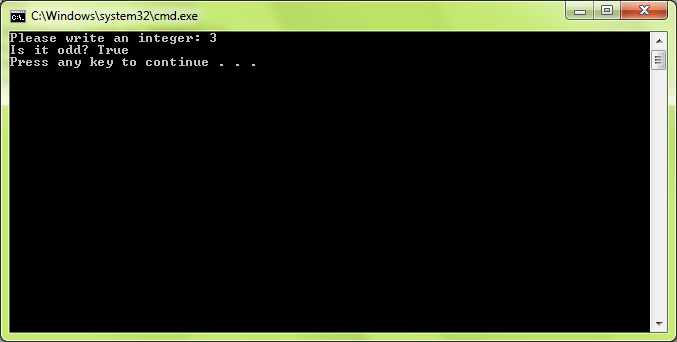
int a = int.Parse(Console.ReadLine());

Console.Write("Is it odd? ");

Console.WriteLine(a % 2 != 0);

}

}



**Problem 2:** 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. Examples:

|  |  |
| --- | --- |
| **weight** | **weight on the Moon** |
| 86 | 14.62 |
| 74.6 | 12.682 |

class MoonGravitation

{

static void Main()

{

Console.Write("Please enter your weight on Earth: ");

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

double gOnEarth = 9.80665;

double gOnMoon = (gOnEarth\*17)/100;

double weightOnMoon = (weightOnEarth \* gOnMoon) / gOnEarth;

Console.WriteLine("Your weight on Moon would be: " + weightOnMoon);

}

}



**Problem 3:** Write a Boolean expression that checks for given integer if it can be **divided** (without remainder) **by 7 and 5 in the same time**. Examples:

|  |  |  |
| --- | --- | --- |
| **n** | **Divided by 7 and 5?** | |
| 3 | false | |
| 35 | true |

class DevidedBySevenAndFive

{

static void Main()

{

Console.Write("Please write an integer: ");

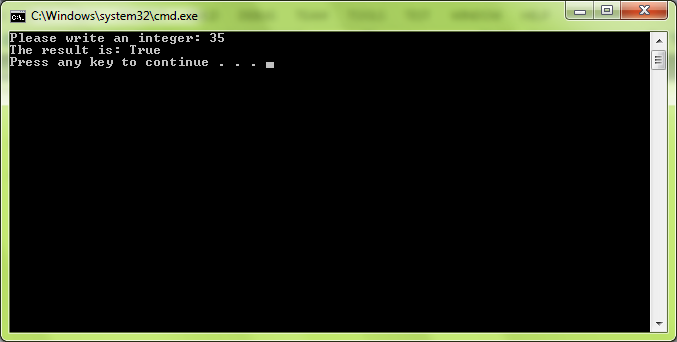
int i = int.Parse(Console.ReadLine());

Console.Write("The result is: ");

Console.WriteLine((i % 5 == 0) & (i % 7 == 0));

}

}



**Problem 4:** Write an expression that calculates **rectangle’s perimeter** and **area** by given **width** and **height**. Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **width** | **height** | **perimeter** | **area** |
| 3 | 4 | 14 | 12 |
| 2.5 | 3 | 11 | 7.5 |

class RectancleAreaAndPerimeter

{

static void Main()

{

Console.Write("Width: ");

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

Console.Write("Height: ");

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

double recArea = a \* b;

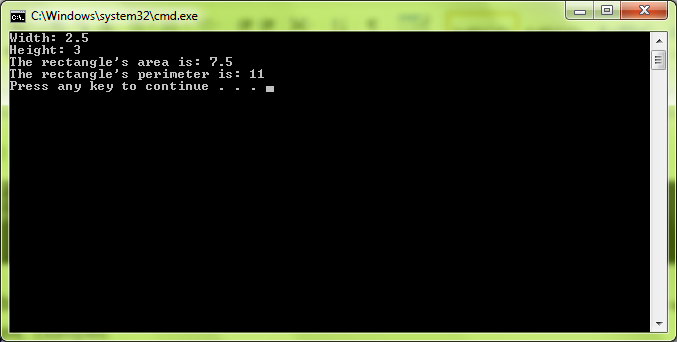
double recPerimeter = 2 \* a + 2 \* b;

Console.WriteLine("The rectangle's area is: " + recArea);

Console.WriteLine("The rectangle's perimeter is: " + recPerimeter);

}

}



**Problem 5:** Write **an expression** that checks for given integer **if its third digit** from right-to-left **is 7**. Examples:

|  |  |
| --- | --- |
| **n** | **Third digit 7?** |
| 5 | false |
| **7**01 | true |

class ThirdDigitIsSeven

{

static void Main()

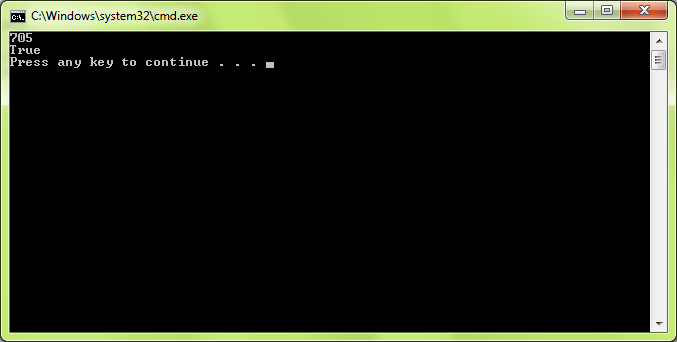
{

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

Console.WriteLine((a/100) % 10 == 7);

}

}



**Problem 6:** Write a program that takes as input a **four-digit number** in format **abcd** (e.g. 2011) and performs the following:

* 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).

The number has always exactly **4 digits** and cannot start with 0. Examples:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **n** | **sum of digits** | **reversed** | **last digit in front** | **second and third digits exchanged** |
| 2011 | 4 | 1102 | 1201 | 2101 |
| 3333 | 12 | 3333 | 3333 | 3333 |

class FourDigitNumber

{

static void Main()

{

Console.WriteLine("Enter a four-digit number, which cannot start with 0: ");

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

int a = fourDigitNum / 1000;

int bHelp = fourDigitNum / 100;

int b = bHelp % 10;

int cHelp = fourDigitNum / 10;

int c = cHelp % 10;

int d = fourDigitNum % 10;

Console.WriteLine(a + b + c + d);

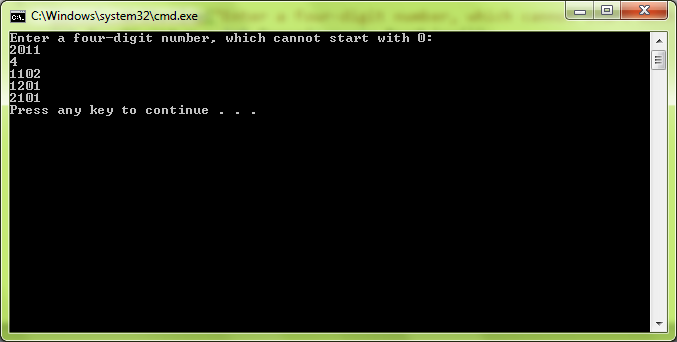
Console.WriteLine("{0}{1}{2}{3}", d, c, b, a);

Console.WriteLine("{0}{1}{2}{3}", d, a, b, c);

Console.WriteLine("{0}{1}{2}{3}", a, c, b, d);

}

}



**Problem 7:** Write **an expression** that checks if given point (**x**, **y**) is inside a **circle K**({**0**, **0**}, **2**). Examples:

|  |  |  |
| --- | --- | --- |
| **x** | **y** | **inside** |
| 0 | 1 | true |
| -2 | 0 | true |
| -1 | 2 | false |

class PointInACircle

{

static void Main()

{

Console.Write("Enter x: ");

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

Console.Write("Enter y: ");

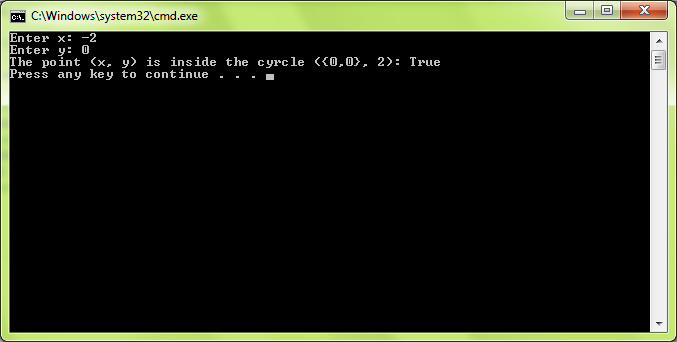
double y = double.Parse(Console.ReadLine());

bool insideTheCyrcle = ((x \* x) + (y \* y) <= 4) && x <= 2 && y <= 2;

Console.WriteLine("The point (x, y) is inside the cyrcle ({0,0}, 2): " + insideTheCyrcle);

}

}



**Problem 8:** Write an **expression** that checks if given positive integer number **n** (**n** ≤ 100) is [prime](https://en.wikipedia.org/wiki/Prime_number) (i.e. it is divisible without remainder only to itself and 1). Examples:

|  |  |
| --- | --- |
| **n** | **Prime?** |
| 1 | false |
| 2 | true |

using System;

class PrimeNumberCheck

{

static void Main()

{

Console.Write("Enter a positive integer number n (n ≤ 100), n = ");

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

bool prime = true;

if (n < 0 || n > 100)

Console.WriteLine("Invalid number");

else if (n < 2)

{

prime = false;

Console.WriteLine(prime);

}

else

{

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

{

if (n % i == 0)

{

prime = false;

}

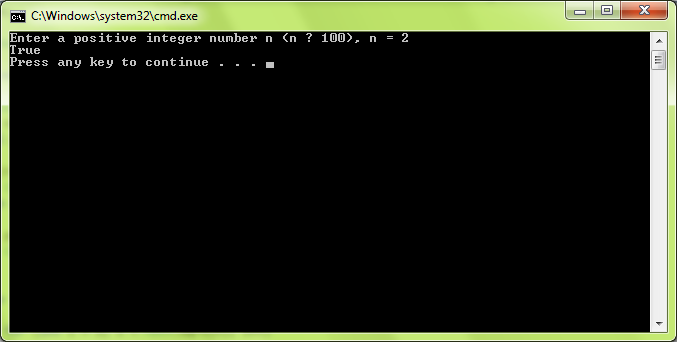
}

Console.WriteLine(prime);

}

}

}



**Problem 9:** Write an expression that calculates **trapezoid's area** by given sides **a** and **b** and height **h**. Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **a** | **b** | **h** | **area** |
| 5 | 7 | 12 | 72 |
| 2 | 1 | 33 | 49.5 |

class AreaOfTrapezoid

{

static void Main()

{

Console.Write("Enter the first base: ");

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

Console.Write("Enter the second base: ");

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

Console.WriteLine("Enter the height of the trapezoid: ");

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

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

Console.WriteLine("The area of the trapezoid is: " + area);

}

}



**Problem 10:** Write an expression that checks for given point (x, y) if it is **within the circle K**({1, 1}, 1.5) and **out of the rectangle R**(top=**1**, left=**-1**, width=**6**, height=**2**). Examples:

|  |  |  |
| --- | --- | --- |
| **x** | **y** | **inside K & outside of R** |
| 1 | 2 | yes |
| 2.5 | 2 | no |

class PointInsideCyrcle

{

static void Main()

{

Console.Write("Enter x: ");

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

Console.Write("Enter y: ");

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

double xAbs = Math.Abs(x - 1);

bool outsideRec = (((x - 1) \* (x - 1) + (y - 1) \* (y - 1) <= 1.5 \* 1.5) && y > 1 && (y-1)<= 1.5 && xAbs <= 1.5);

Console.Write("Is the point inside the circle and outside the rectangle? ");

if (outsideRec == true)

{

Console.WriteLine("yes");

}

else

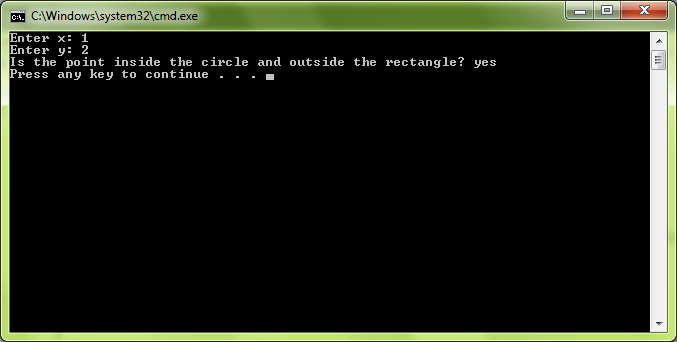
{

Console.WriteLine("no");

}

}

}



**Problem 11:** Using bitwise operators, write an **expression** for finding the value of the bit #**3** of a given unsigned integer. The bits are counted from right to left, starting from bit #0. The result of the expression should be either **1 or 0**. Examples:

|  |  |  |
| --- | --- | --- |
| **n** | **binary representation** | **bit #3** |
| 5 | 00000000 0000**0**101 | 0 |
| 0 | 00000000 0000**0**000 | 0 |

class ExtractBit3

{

static void Main()

{

Console.Write("Enter an integer n = ");

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

int p = 3;

int nRightP = n >> p;

int bit = nRightP & 1;

Console.WriteLine("The third bit of the integer {0} is {1}", n, bit);

}

}



**Problem 12:** Write an expression that extracts from given integer **n** the value of given **bit at index** **p**. Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **binary representation** | **p** | **bit @ p** |
| 5 | 00000000 00000**1**01 | 2 | 1 |
| 0 | 000000**0**0 00000000 | 9 | 0 |

class ExtractBitFromInteger

{

static void Main()

{

Console.Write("Enter an integer n = ");

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

Console.Write("Enter an index p = ");

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

int nRightP = n >> p;

int bit = nRightP & 1;

Console.WriteLine("The value of the index {0} is {1} ", p, bit);

}

}



**Problem 13:** Write a **Boolean expression** that returns if the **bit at position p** (counting from **0**, starting from the right) in given integer number **n** has value of **1**. Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **binary representation of n** | **p** | **bit @ p == 1** |
| 5 | 00000000 00000**1**01 | 2 | true |
| 0 | 000000**0**0 00000000 | 9 | false |

class CheckingBit

{

static void Main()

{

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

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

int nRightP = n >> p;

int bit = nRightP & 1;

bool hasValue1 = (bit == 1);

Console.WriteLine(hasValue1);

}

}

