# Homework: Operators Expressions and Statements

This document defines homework assignments from the [“C# Basics“ Course @ Software University](http://softuni.bg/courses/csharp-basics/). Please submit as homework a single zip / rar / 7z archive holding the solutions (source code only) of all below described problems.

## Odd or Even Integers

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

|  |  |
| --- | --- |
| **n** | **Odd?** |
| 3 | true |
| 2 | false |
| -2 | false |
| -1 | true |
| 0 | false |

## Gravitation on the Moon

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 |
| 53.7 | 9.129 |

## Divide by 7 and 5

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 |
| 0 | false |
| 5 | false |
| 7 | false |
| 35 | true |
| 140 | true |

## Rectangles

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 |
| 5 | 5 | 20 | 25 |

## Third Digit is 7?

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 |
| 9**7**03 | true |
| **8**77 | false |
| 777**8**77 | false |
| 9999**7**99 | true |

## Four-Digit Number

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 |
| 9876 | 30 | 6789 | 6987 | 9786 |

## Point in a Circle

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 |
| 1.5 | -1 | true |
| -1.5 | -1.5 | false |
| 100 | -30 | false |
| 0 | 0 | true |
| 0.2 | -0.8 | true |
| 0.9 | -1.93 | false |
| 1 | 1.655 | true |

## Prime Number Check

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 |
| 3 | true |
| 4 | false |
| 9 | false |
| 97 | true |
| 51 | false |
| -3 | false |
| 0 | false |

## Trapezoids

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 |
| 8.5 | 4.3 | 2.7 | 17.28 |
| 100 | 200 | 300 | 45000 |
| 0.222 | 0.333 | 0.555 | 0.1540125 |

## Point Inside a Circle & Outside of a Rectangle

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 |
| 0 | 1 | no |
| 2.5 | 1 | no |
| 2 | 0 | no |
| 4 | 0 | no |
| 2.5 | 1.5 | NO! |
| 1 | 2.5 | yes |
| -100 | -100 | no |

## Bitwise: Extract Bit #3

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 |
| 15 | 00000000 0000**1**111 | 1 |
| 5343 | 00010100 1101**1**111 | 1 |
| 62241 | 11110011 0010**0**001 | 0 |

## Extract Bit from Integer

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 |
| 15 | 00000000 000011**1**1 | 1 | 1 |
| 5343 | 00010100 **1**1011111 | 7 | 1 |
| 62241 | 1111**0**011 00100001 | 11 | 0 |

## Check a Bit at Given Position

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 |
| 15 | 00000000 000011**1**1 | 1 | true |
| 5343 | 00010100 **1**1011111 | 7 | true |
| 62241 | 1111**0**011 00100001 | 11 | false |

## Modify a Bit at Given Position

We are given an integer number **n**, a bit value **v** (v=0 or 1) and a position **p**. Write a **sequence of operators** (a few lines of C# code) that modifies **n** to hold the value **v** at the position **p** from the binary representation of **n** while preserving all other bits in **n**. Examples:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **n** | **binary representation of n** | **p** | **v** | **binary result** | **result** |
| 5 | 00000000 00000**1**01 | 2 | 0 | 00000000 00000**0**01 | 1 |
| 0 | 000000**0**0 00000000 | 9 | 1 | 000000**1**0 00000000 | 512 |
| 15 | 00000000 000011**1**1 | 1 | 1 | 00000000 000011**1**1 | 15 |
| 5343 | 00010100 **1**1011111 | 7 | 0 | 00010100 **0**1011111 | 5215 |
| 62241 | 1111**0**011 00100001 | 11 | 0 | 1111**0**011 00100001 | 62241 |

## \* Bits Exchange

Write a program that **exchanges bits** **3**, **4** and **5** with bits **24**, **25** and **26** of **given 32-bit unsigned integer**. Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **binary representation of n** | **binary result** | **result** |
| 1140867093 | 01000**100** 00000000 01000000 00**010**101 | 01000**010** 00000000 01000000 00**100**101 | 1107312677 |
| 255406592 | 00001**111** 00111001 00110010 00**000**000 | 00001**000** 00111001 00110010 00**111**000 | 137966136 |
| 4294901775 | 11111**111** 11111111 00000000 00**001**111 | 11111**001** 11111111 00000000 00**111**111 | 4194238527 |
| 5351 | 00000**000** 00000000 00010100 11**100**111 | 00000**100** 00000000 00010100 11**000**111 | 67114183 |
| 2369124121 | 10001**101** 00110101 11110111 00**011**001 | 10001**011** 00110101 11110111 00**101**001 | 2335569705 |

## \*\* Bit Exchange (Advanced)

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. The first and the second sequence of bits may **not overlap**. Examples:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **n** | **p** | **q** | **k** | **binary representation of n** | **binary result** | **result** |
| 1140867093 | 3 | 24 | 3 | 01000**100** 00000000 01000000 00**010**101 | 01000**010** 00000000 01000000 00**100**101 | 1107312677 |
| 4294901775 | 24 | 3 | 3 | 11111**111** 11111111 00000000 00**001**111 | 11111**001** 11111111 00000000 00**111**111 | 4194238527 |
| 2369124121 | 2 | 22 | 10 | **10001101 00**110101 1111**0111 000110**01 | **01110001 10**110101 1111**1000 110100**01 | 1907751121 |
| 987654321 | 2 | 8 | 11 | - | - | overlapping |
| 123456789 | 26 | 0 | 7 | - | - | out of range |
| 33333333333 | -1 | 0 | 33 | - | - | out of range |