# Homework: Conditional 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.

## Exchange If Greater

## Bonus Score

Write a program that applies bonus score to given score in the range [1…9] by the following rules:

* If the score is between 1 and 3, the program multiplies it by 10.
* If the score is between 4 and 6, the program multiplies it by 100.
* If the score is between 7 and 9, the program multiplies it by 1000.
* If the score is 0 or more than 9, the program prints “invalid score”.

Examples:

|  |  |
| --- | --- |
| **score** | **result** |
| 2 | 20 |
| 4 | 400 |
| 9 | 9000 |
| -1 | invalid score |
| 10 | invalid score |

## Check for a Play Card

Classical play cards use the following signs to designate the card face: 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K and A. Write a program that enters a string and prints “yes” if it is a valid card sign or “no” otherwise. Examples:

|  |  |
| --- | --- |
| **character** | **Valid card sign?** |
| 5 | yes |
| 1 | no |
| Q | yes |
| q | no |
| P | no |
| 10 | yes |
| 500 | no |

## Multiplication Sign

Write a program that shows the sign (+, - or 0) of the product of three real numbers, without calculating it. Use a sequence of **if** operators. Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **a** | **b** | **c** | **result** |
| 5 | 2 | 2 | + |
| -2 | -2 | 1 | + |
| -2 | 4 | 3 | - |
| 0 | -2.5 | 4 | 0 |
| -1 | -0.5 | -5.1 | - |

## The Biggest of 3 Numbers

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

## The Biggest of Five Numbers

## Sort 3 Numbers with Nested Ifs

Write a program that enters **3 real numbers** and prints them sorted in descending order. Use nested **if** statements. Don’t use arrays and the built-in sorting functionality. Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **a** | **b** | **c** | **result** |
| 5 | 1 | 2 | 5 2 1 |
| -2 | -2 | 1 | 1 -2 -2 |
| -2 | 4 | 3 | 4 3 -2 |
| 0 | -2.5 | 5 | 5 0 -2.5 |
| -1.1 | -0.5 | -0.1 | -0.1 -0.5 -1.1 |
| 10 | 20 | 30 | 30 20 10 |
| 1 | 1 | 1 | 1 1 1 |

## Digit as Word

## Play with Int, Double and String

Write a program that, depending on the user’s choice, inputs an **int**, **double** or **string** variable. If the variable is **int** or **double**, the program increases it by one. If the variable is a **string**, the program appends "**\***" at the end. Print the result at the console. Use **switch** statement. Example:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **program** | **user** |  | **program** | **user** |
| Please choose a type: 1 --> int  2 --> double  3 --> string | 3 | Please choose a type: 1 --> int  2 --> double  3 --> string | 2 |
| Please enter a string: | hello | Please enter a double: | 1.5 |
| hello\* |  | 2.5 |  |

## \* Beer Time

A beer time is after 1:00 PM and before 3:00 AM. Write a program that **enters a time** in format “hh:mm tt” (an hour in range [01...12], a minute in range [00…59] and AM / PM designator) and prints “**beer time**” or “**non-beer time**” according to the definition above or “**invalid time**” if the time cannot be parsed. Note that you may need to learn how to parse dates and times. Examples:

|  |  |
| --- | --- |
| **time** | **result** |
| 1:00 PM | beer time |
| 4:30 PM | beer time |
| 10:57 PM | beer time |
| 8:30 AM | non-beer time |
| 02:59 AM | beer time |
| 03:00 AM | non-beer time |
| 03:26 AM | non-beer time |

## \* Number as Words

Write a program that **converts a number in the range [0…999] to words**, corresponding to the English pronunciation. Examples:

|  |  |
| --- | --- |
| **numbers** | **number as words** |
| 0 | Zero |
| 9 | Nine |
| 10 | Ten |
| 12 | Twelve |
| 19 | Nineteen |
| 25 | Twenty five |
| 98 | Ninety eight |
| 273 | Two hundred and seventy three |
| 400 | Four hundred |
| 501 | Five hundred and one |
| 617 | Six hundred and seventeen |
| 711 | Seven hundred and eleven |
| 999 | Nine hundred and ninety nine |

## \* Zero Subset

We are given 5 integer numbers. Write a program that finds all **subsets of these numbers whose sum is 0**. Assume that repeating the same subset several times is not a problem. Examples:

|  |  |
| --- | --- |
| **numbers** | **result** |
| 3 -2 1 1 8 | -2 + 1 + 1 = 0 |
| 3 1 -7 35 22 | no zero subset |
| 1 3 -4 -2 -1 | 1 + -1 = 0  1 + 3 + -4 = 0  3 + -2 + -1 = 0 |
| 1 1 1 -1 -1 | 1 + -1 = 0  1 + 1 + -1 + -1 = 0  1 + -1 + 1 + -1 = 0  … |
| 0 0 0 0 0 | 0 + 0 + 0 + 0 + 0 = 0 |

Hint: you may check for zero each of the 32 subsets with 32 if statements.