**CISS240: Introduction to Programming**
**Assignment 5**

Name: _____

Objectives

1. Declare boolean variables.
2. Use boolean operators to form boolean expressions.
3. Perform output for boolean expressions/values/variables.

Q1. Write a program that prompts the user for three integers for the lengths of a triangle and display whether the three integers form the sides of a right-angled triangle: it will display `1` if the three sides represents a right-angled triangle and `0` otherwise. The third integer entered is the largest of the three.

Test 1.

```
3 4 5
1
```

Test 2.

```
3 4 6
0
```

[Hint: Let $a$, $b$, $c$ be the length of the sides of a right-angled triangle ($c$ being the longest). If the sum of the squares of $a$ and $b$ equals the square of $c$, then the triangle is a right-angled triangle. You need to translate the above to a C++ boolean expression. This is the Pythagorean theorem. Your program should print the boolean value of whether $a$, $b$ and $c$ form a right-angled triangle. Recall that the `true` boolean value when printed is `1` and `false` is printed as `0`. This is exactly what the above requires. You can either print the boolean value of a boolean expression:

```
std::cout << [some boolean expression] << '\n';
```

or you can store the boolean value in a boolean variable (with a reasonable name) and print the value of the variable:

```
bool is_rt_angle_triangle = [some boolean expression];
std::cout << is_rt_angle_triangle << '\n';
```

If your boolean expression is long, you might want to break it up into smallest boolean expressions and then join them up using and operator `&&` or the or operator `!!`.]

Q2. Write a program that prompts the user for three integers and print `1` if the integers are in ascending order and `0` otherwise. Note that `1 1 1` is in ascending order while `2 3 1` is not (because of the last integer).

Test 1.

```
10 11 12
1
```

Test 2.

```
10 10 11
1
```

Test 3.

```
10 11 11
1
```

Test 4.

```
11 11 11
1
```

Test 5.

```
11 11 10
0
```

Test 6.

```
11 10 11
0
```

Test 7.

```
10 11 11
1
```

Test 8.

```
11 10 9
0
```

Q3. An intercept missile is capable of exploding and disabling an enemy missile if it is sufficiently close to the latter. The position $(x, y, z)$ of the enemy missile is continually collected by a radar mounted on the intercept missile. The distance between two points $(x, y, z)$ and $(0, 0, 0)$ (the position of the intercept missile) in 3–dimensions is given by the formula

$$\sqrt{x^2 + y^2 + z^2}$$

(3–dimensional Pythagorus formula). Write a program that detects whether the distance from the enemy missile to the intercept missile is sufficiently close for the intercept missile to explode. Your program accepts 4 doubles: The first three values gives the $(x, y, z)$ position of the enemy missile from the intercept missile and the last is the maximum distance the intercept should explode to disable the enemy missile. Your program must set a boolean variable to true exactly when the intercept missile should explode and then print this boolean variable.

Test 1.

```
2 0 0 1
0
```

Test 2.

```
2 0 0 2.5
1
```

Test 3.

```
1 0 0 1
1
```

Test 4.

```
1 1 0 1.5
1
```

Test 5.

```
1.1 1.2 1.3 2
0
```

Q4. Write a program that prompts the user for 4 integers that represent time, sets a boolean variable to true exactly when the time entered is valid, and then prints the boolean variable. The 4 integer values are as follows:

- The first integer is 0 if the time entered is in 12–hour format; it is 1 if the time entered is in 24–hour format
- The second integer represents the hour. Note that in the 12–hour format, the hour is valid if it is from 1 to 12 (inclusive); in the 24–hour format, the hour is valid if it is from 0 and 23 (inclusive).
- The third integer represents the minute.
- The fourth integer represents the second.

Test 1.

```
0 -1 -5 -1
0
```

Test 2.

```
0 1 -5 2
0
```

Test 3.

```
0 1 70 2
0
```

Test 4.

```
0 1 7 244
0
```

Test 5.

```
0 0 7 2
0
```

Test 6.

```
0 0 13 2
0
```

Test 7.

```
0 1 7 2
1
```

Test 8.

```
1 1 -5 1
0
```

Test 9.

```
1 1 -5 2
0
```

Test 10.

```
1 1 70 2
0
```

Test 11.

```
1 1 7 244
0
```

Test 12.

```
1 0 7 2
1
```

Test 13.

```
1 0 13 2
1
```

Test 14.

```
1 24 7 2
0
```

Q5. The following is not a program. You need to write your work to a text file. You can use Notepad (on Windows 7 or 8) or some other text editor (such as emacs/xemacs in Linux) and save the file with the name `a05q05.txt`. (Talk to me or any of the senior CS tutors if you have never used a text editor before.)

There is a *boolean* expression below. You need to evaluate it one operator at a time according to the standard precedence and associative rules until you obtain a boolean value. Note that type casting is considered a step. There is also a rule below on removing parentheses. Here's how you should present your work in the text file. For the expression `1 + 2 < 3 * 4`, if you believe the `+` goes first, followed by the `<`, and then the `*`, you write your answer like this in the file:

```
// Name: smaug
// File: a05q05.txt

1 + 2 < 3 * 4
= 3 < 3 * 4                                    by 1 + 2 = 3
= false * 4                                    by 3 < 3 = false
= 0 * 4                                        by int(false) = 0
= 0                                            by 0 * 4 = 0
= false                                        by bool(0) = false
```

Note that, in the above, there are <u>two</u> type casts.

Of course the above is incorrect. The point is just to show you how to present your work.

You need not follow the exactly column placement of the reasoning that starts with "by ...". For instance, see another example below where the placement is different.

Now I will show you how to remove parentheses. If you believe for the following that the parenthesized expression should be evaluated first, followed by the `&&`, and finally by the `||`, this is how you must present your work:

```
// Name: smaug
// File: a05q05.txt

true && (2 < 3) || false
= true && (true) || false                  by 2 < 3 = true
= true && true || false                    by (true) = true
= true || false                            by true && true = true
= true                                     by true || false = true
```

Note that the removal of the parentheses is a step of its own; look at the above computation very carefully.

Now for the question:

Evaluate the following expression according to the above requirements:

```
true || (2 + 3 < 5 - 1) && !(3 / 4 >= (5 + (2 - 1) * 2)) || (2 < 1)
```

Grading. The grading of this question is very strict: make sure you follow the above rules. Furthermore, grading will stop once an error occurs. For instance if you have typed 20 lines in your file and the first time the grader discovers an error is at the 18–th line, then your score is equivalent to 17 out of 20. Check your work very carefully. Extraneous spaces are removed. For instance the grader will automatically change this file

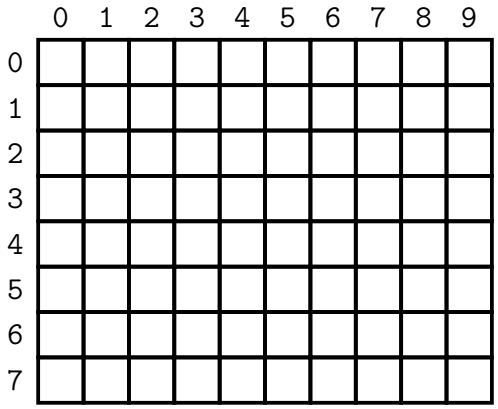
```
// Name: smaug
// File: a05q05.txt

1 + 2 < 3 * 4
= 3 < 3 * 4                                    by 1 + 2 = 3
= false * 4                                    by 3 < 3 = false
= 0 * 4                                        by int(false) = 0
= 0                                            by 0 * 4 = 0
= false                                        by bool(0) = false
```

to this:

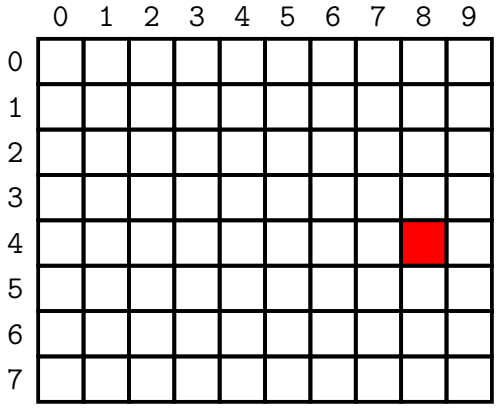
```
// Name: smaug
// File: a05q05.txt

1 + 2 < 3 * 4
= 3 < 3 * 4 by 1 + 2 = 3
= false * 4 by 3 < 3 = false
= 0 * 4 by int(false) = 0
= 0 by 0 * 4 = 0
= false by bool(0) = false
```

You should still present your work with the word "`by`" aligned so that it's easier for you to check your work.

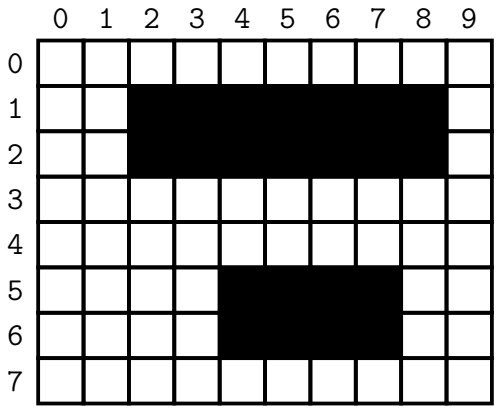
Q6. You are writing a tile–based dungeon–type game. The dungeon is made up of a grid of rooms. A player can go from a room to an adjoining room. Each room is labeled with a row number and a column number. Here's the overall plan of the dungeon:




Note that the row numbers are from 0 to 7 and the column numbers are from 0 to 9. A player in the dungeon, of course, must stay in the dungeon. The player's position is given by the `row` and `col` variable (obviously of `int` type). For instance, in the following state of the game, the player is at `row = 4` and `col = 8`.




In addition, there are special rooms which are closed and the person cannot enter these rooms (without special spells.) Here are the special rooms:




At the beginning of the game, you have to place a person in a regular (i.e., non–special) room. Your goal is to write a program that gets the position (i.e., values for `row` and `col` in that order) and prints `1` if the position corresponds to a regular room. Otherwise (i.e., the room is special or the position is outside the dungeon), your program prints `0`. You should use the simplest boolean expression.

Three test cases are included (only to show you the required input/output session with the program.) You are strongly advised to include more test cases.

Test 1.

```
0 8
1
```

Test 2.

```
5 7
0
```

Test 3.

```
100 100
0
```