- No late tutorials will be accepted.
- If there are specific instructions for making a function, please follow them exactly. That means that
  - function names
  - function return types
  - parameter types and order

  should all be **EXACTLY** as described. If the script can't read it, you will receive 0 for that part.
- Your **Tutorial 3** code will be marked by a Python script. You are being given the script so that you may make sure your code runs correctly. As such, submissions with improper files, configuration, or function signatures will not be accepted.

1 Download and Submission Instructions

Download `tutorial3.zip` and unzip it into your workspace. You will find the test script `t3test.py` and a `tutorial3` folder. In the `tutorial3` folder you will find `test1.cc`, `test2.cc` and `Makefile`. Files `test1.cc` and `test2.cc` are copies of the files that will be used to test your tutorial. When your tutorial is marked they will be replaced with fresh copies. The `Makefile` is a suggested starting point. You may make any changes to it you wish, as long as `make all` makes `test1` and `test2` executables, and `make clean` removes all executables and object files.

You should add to the `tutorial3` folder the header and source files for two classes, `Room` and `Date`, from Assignment 1. Once you have written the classes and completed your tests, you should zip the `tutorial3` directory into a file `tutorial3.zip` and submit it to Brightspace. If you are zipping the files in the course VM you must zip them from the command line. Open a terminal in the folder that contains "tutorial3". Use the command `zip tutorial3.zip tutorial3`. This will zip the `tutorial3` folder, or update it if you change the contents. Submit `tutorial3.zip` to Brightspace by the deadline. DO NOT USE .tar OR .tar.gz FILES. Use .zip only please.

2 Testing Your Tutorial With `t3test.py`

`t3test.py` is a test script that is very similar to what will be used to mark your tutorial (basically I will change the input and expected output ... unless I get lazy, then I won't change anything). So the mark you see here should be the mark you receive (as long as you did not hard code output). To run `t3test.py`, paste or save this file in the directory that contains your "tutorial3" folder. Open a command line. You may have to make it executable, so type `chmod +x t3test.py`. You may run the script as is, in which case it will look for a file to unzip. Or, if you have not zipped your files yet you may supply a "-nozip" argument, in which case it looks for the `tutorial3` folder.

To have the script unzip `tutorial3.zip` and then test your code, run `./t3test.py`. To skip the unzip step use `./t3test.py -nozip`. When your tutorial is being officially marked we expect a zipped file.

Running this script will generate a file "results.txt" just outside of the "tutorial3" folder. This will have some useful output as well as the mark.

3 Learning Outcomes

This tutorial will test the application logic in two of the classes from Assignment 1.

4 Instructions

4.1 Overview

In this tutorial you will write the `Room` (Section 5.2) and `Date` (Section 5.3) classes from Assignment 1. There are two test files provided, `test1.cc` and `test2.cc` that will test the `overlaps` and `meetsCriteria` functions from those classes. You must provide a Makefile that compiles `Date` and `Room` into object files and compiles `test1.cc` and `test2.cc` into executables linking to the appropriate object file. As usual the test script, `t3test.py` is provided.

4.2 Room Class

Complete Section 5.2 in Assignment 1. Link the object file `Room.o` to the `test1` executable.

4.3 Date Class

Complete Section 5.3 in Assignment 1. Link the object file `Date.o` to the `test2` executable.

4.4 Makefile

Your Makefile should compile two object files, `Room.o` and `Date.o`. It should link `Room.o` to the `test1` executable and `Date.o` to the `test2` executable. For example, assuming you have compiled `Room.o`, you can compile the `test1` executable by including the following in your Makefile:

```
test1:  test1.cc Room.o
    g++ -o test1 test1.cc Room.o
```

In addition your Makefile should contain an `all` command that creates the `test1` and `test2` executables and a `clean` command that removes all executables and object files.

4.5 t3test.py

Run this python script to test the functions described above from the `Room` and `Date` classes.