
- 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 5** 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 Submission Instructions

Download `tutorial5.zip`. Unzip it into your working directory. There is a directory `tutorial5` and the test file `t5test.py`. In the `tutorial5` folder are `test.cc`, `defs.h`, and `Makefile` to get you started. To the `tutorial5` folder you should add the following files.

1. Header and source files for the `Episode` class from Assignment 2, Section 6.2.
2. Header and source files for the `Podcast` class from Assignment 2, Section 6.3.

You will zip the `tutorial5` directory into a file `tutorial5.zip`. If you are doing this in the course VM you must do this from the command line. Open a terminal in the folder that contains `tutorial5`. Use the command `zip -r tutorial5.zip tutorial5`. This will zip the `tutorial5` folder, or update it if you change the contents. Submit `tutorial5.zip` to Brightspace by the deadline. DO NOT USE .tar OR .tar.gz FILES. Use .zip only please.

2 Testing Your Tutorial With `t5test.py`

`t5test.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. To run `t5test.py`, open a command line in the directory containing `t5test.py`. You may have to make it executable, so type `chmod +x t5test.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 `tutorial5` folder.

To have the script unzip `tutorial5.zip` and then test your code, run `./t5test.py`. To skip the unzip step use `./t5test.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 `tutorial5` folder. This will have some useful output as well as the mark.

3 Learning Outcomes

This tutorial will test the deep-copy capabilities of the `Podcast` copy constructor.

4 Instructions

4.1 Overview

In this tutorial you will write the `Podcast` (Section 6.3) class from Assignment 2 (and, if you are not done yet, the `Episode` class in Section 6.2). There is one test file provided, `test.cc` that will test the functions that you provide from those classes. You are provided with a Makefile - make any changes you see fit (and remember that Linux is case sensitive!). As usual the test script, `t5test.py` is provided. This script is run with `valgrind`, so it will check the `valgrind` output to see if there are memory leaks.

4.2 Episode Class

Complete Section 6.2 in Assignment 2 if you have not done so already. Include the `Episode` header and source files in your `tutorial5` folder.

4.3 Podcast Class

Complete Section 6.3 of assignment 2. Include the `Podcast` header and source files in the `tutorial5` folder.

4.4 Makefile

A Makefile is provided for you, but you may change it or use your own. Your Makefile should compile two object files, `Episode.o` and `Podcast.o`. It should link these object files to the `test` executable. In addition your Makefile should contain an `all` command that creates the `test` executable and a `clean` command that removes all executables and object files.

4.5 t5test.py

Run this python script to test the functions described above. Correct all errors.