
- 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 10** 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 "tutorial10.zip". Unzip it into your working directory. There is a directory "tutorial10" and the test file "t10test.py". In the "tutorial10" folder are "Student.h", "Student.cc", "TestClass.h", "TestClass.cc","Algorithms.h", "test.cc" and "Makefile". In this tutorial you will implement the "Algorithms.cc" file.

Once you have written the classes and completed your tests, submit this tutorial to Gradescope. The Gradescope server is flexible in how you submit. You may zip the folder, zip the files, or submit the folder itself or individual files. What follows are one set of instructions that will work on Gradescope. You will zip the "tutorial10" directory into a file "tutorial10.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 "tutorial10". Use the command `zip -r tutorial10.zip tutorial10`. This will zip the `tutorial10` folder, or update it if you change the contents. DO NOT USE tar FILES. These do not seem to work well with Gradescope. Submit to Gradescope by the deadline. You will receive your mark immediately, and you may submit as many times as you like. You may also submit tutorials up to one week late for a 10% penalty.

2 Testing Your Tutorial With `t10test.py`

`t10test.py` is a test script that is very similar to the script that is being run on Gradescope (there might be slightly different input or different even tests). So the mark you see here should be close to the mark you will receive on Gradescope. To run `t10test.py`, open a command line in the directory containing `t10test.py`. You may have to make it executable, so type `chmod +x t10test.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 "tutorial10" folder.

To have the script unzip `tutorial10.zip` and then test your code, run `./t10test.py`. To skip the unzip step use `./t5test.py -nozip`. When your tutorial is being officially marked you SHOULD NOT SUBMIT A .tar FILE. Either submit a zipped file or drag and drop the tutorial10 folder into Gradescope.

Running this script will generate a file "results.txt" just outside of the "tutorial10" folder. This will have some useful output as well as the mark.

3 Learning Outcomes

In this tutorial you will learn some of the STL library including `vectors`, `algorithms`, and `iterators`.

4 Instructions

4.1 Overview

In this tutorial you are to implement an `Algorithms` class. This class will read in a collection `Student` objects and perform different operations on them such as sorting.

4.2 Student Class

This class is provided for you. However, you may add functions to it as needed (and you almost certainly will). You may also need to make it a `friend` to some of your (global) functions to properly implement your algorithms. You may also make it a `friend` to your `Algorithms` class if you desire. In short, make it work using the tools at your disposal.

4.3 Algorithms Class

The `Algorithms.h` file has been provided. You must implement the functions shown. In addition you may add member variables or helper functions as you see fit. You should also write the `Algorithms.cc` file. This class will hold multiple `Student` objects and run algorithms on them. You may use any data structure you wish to store the `Students` (though `vector` is recommended). The functions are specified as follows:

1. `void addStudent(const string& number, const string& name, float);`
   Add a new `Student` object to your data structure.
2. `void getPassingStudents(vector<Student>& v);`
   `vector<Student>& v` is a `vector` that is passed in to your function. You should copy or add all passing `Students` in your data structure to `v`. A `Student` passes if their `gpa >= 6.0`.
3. `void sortByNumber(vector<Student>& v);`
   `vector<Student>& v` is a `vector` that is passed in to your function. You should copy or add all `Students` in your data structure to `v` in sorted order by `Student number` (increasing order).
4. `void sortByName(vector<Student>& v);`
   `vector<Student>& v` is a `vector` that is passed in to your function. You should copy or add all `Students` in your data structure to `v` in sorted order by `Student name` (increasing order).
5. `bool highestGpa(vector<Student>::iterator& stuIt);`
   Assign an iterator to `stuIt` that contains the `Student` with the highest `gpa`. When assigning the iterator, you may use the assignment operator (as in `stuIt = theIterator;`).
6. `bool findStudent(const string& name, vector<Student>::iterator& stuIt);`
   If a `Student` with the name `name` exists, assign the iterator containing the `Student` to `stuIt` and return true. If no such `Student` is found, return false.

4.4 TestClass Class

This class contains static functions used to test your `Algorithms` class. You may modify it for testing purposes (such as adding `cout` statements), but it is recommended you do not change the functionality. When your tutorial is marked, new copies of `TestClass` will be copied in and used.

4.5 Makefile

A Makefile is provided. You may make changes if you wish. Any Makefile you provide should contain an `all` command that creates the `test` executable and a `clean` command that removes all executables and object files.

4.6 t10test.py

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