# *Computing Science 1P*

# Laboratory Examination 2    2011-12

**16-18 April 2012**  **Time allowed:** 1 hour 50 minutes + advance preparation

**Weight:** the exam contributes 10% to the overall assessment for the CS1P course.

**Threshold:** students must attempt at least 75% of all assessments on the CS1P course in order to gain credit for the course. This exam contributes 10%.

## Instructions

### Before the exam

**You should prepare a complete solution to the exam question given in this pack in advance of the exam**. Put in as many hours as you need to get a properly working solution **that you fully understand**. You can of course type it into a machine and thoroughly test and debug it.

When you come into the exam, you will be able to bring nothing with you. We will provide you with another copy of the question along with some rough paper to work on. You need to be prepared to reconstruct your solution to the problem during the exam.

You may of course talk to others while preparing for the exam. Remember, however, that you will be on your own in the exam, and so you must understand your solution *thoroughly* before coming to the exam.

During the exam, the machines will be disconnected from the network and from your filestores. Note however that the Python Docs help in the Idle Help menu will be available. You should familiarise yourself with its contents, in case you get stuck during the exam with Python syntax.

There will no access to the laboratory, other than to take the lab exam, at any time during the period 16-18 April inclusive. The lab may also be closed for short periods during the week before to allow for preparation by the systems team.

### Starting the exam

Complete the attendance slip and place it in front of you ***together with your matriculation/student ID card***.

Log in to the system using your special exam username and password as provided on the document in your examination pack. Use AMS to set up `LabExam2`. This will generate a copy of the template program file, `Final Solution`, in your exam account Workspace.

### During the exam

As in a normal examination, communication between candidates during the laboratory exam is strictly forbidden.

A candidate who wishes to print a document during the exam should summon an invigilator, who will collect it from the printer and deliver it.

A candidate may leave early, but only after summoning an invigilator, who will ensure that submission has taken place. Any candidate who experiences a hardware or software problem during the examination should summon an invigilator at once.

### At the end of the exam

Candidates should ensure that they have submitted their solution, using AMS, before the end of the examination.

Candidates must leave the laboratory quietly; the exam may still be in progress for other groups.

## Lab Exam Problem: The Swimming Program

The problem is to write a program to help a swimming club to keep track of the best times (or personal bests) for their swimmers. They need to record the event that the swimmer competed in – an event can be one of the following:

- Individual Medley (IM)
- Breast stroke (BS)
- Backstroke (BA)
- Crawl (CR)
- Freestyle (FS)

They also need to record the name of the swimmer, their age (after their last birthday – no need to record days or months), their gender (M/F) and the time (recorded in seconds and tenths of a second). This information is held in a file, `test_data.txt`, also in the `LabExam2` on AMS. This consists of lines with the following format:

```
Event:Gender:Age:Time:Name
```

For example, the four personal best times might be represented by the following lines.

```
IM:F:6:58.2:Kirsty Laing
FS:F:6:44.0:Joan Pine
IM:M:6:57.2:Eric Idle
IM:M:6:58.2:Ciaran Johnson
```

All interaction with the program should be based on text via the Python shell. In other words, there is no need to implement a GUI. However, the user should be prompted to select from a menu as follows:

```
Choose from the following options:
l:   load results from a file
s:   save results to a file
p:   print results
c:   check and add a new personal best

q:   quit
Enter your choice:
```

The following paragraphs provide further information about what each of these options should do.

**1. Loading and Saving a File**

Your program must be able to read information about previous best times from a text file in the format given above if the user selects 'l' from the previous menu. Your solution must also be able to save the information about any swims back to a file in the same text format. The user must be able to specify the name of the files in both cases. (Hint: make sure that you can read back in a file that you have saved to test both functions work as intended).

## 2. Printing the Swimming Information

If the use chooses to print personal bests, this should be displayed to the screen using the following format.

```
Time Event Gender  Age Name
44.0   FS   F       6    Joan Pine

Time Event Gender  Age Name
58.2   IM   F       6    Kirsty Laing

Time Event Gender  Age Name
57.2   IM   M       6    Eric Idle
58.2   IM   M       6    Ciaran Johnson
```

A key requirement of the program is that for each event and age group the times should be sorted so that the fasted result is printed first – as can be seen above where Eric Idle is faster that Ciaran Johnson.

## 3. Checking for and Adding a Personal Best

If a swimmer competes in a race then your program must be able to check if they have achieved a new personal best by searching in the records to see if the new time is faster than any time for the swimmer in an event and age group  (Hint: what happens if two swimmers with the same name compete in the same event?).

## Test Data

The AMS folder includes an input file, which you can use for testing.

# Specific Instructions

## What you must do and what you must submit

You **may** use any standard Python modules, functions and methods. The `split` function from the `string` module (alternatively, the `split` method of a string), is likely to be useful.

You **must** do your work in the file **Final_Solution.py**, which is provided as part of the AMS setup.

## Suggestions on how to progress

1. Design a data structure and write code to process an input file into the data structure.
2. Write the function to save the data structure.
3. Write the function to print all swimming times
4. Write the function to check for and if necessary add a new personal fastest time.

## Marking guidelines

The exam will be marked out of 20. There are 2 marks for the main data structure, 2 for layout and comments, 2 for identifying improvements to your code, 4 marks for file handling, 3 marks for printing delivery information 3 marks for adding a product and 5 marks for identifying the products that are after the best before date. Marks will be awarded up to a maximum of 20.

The following aspects of the program will be taken into account in marking:

- use of appropriate and correct algorithms and Python control structures
- use of appropriate data structures
- correct Python syntax
- good programming style, including layout, use of explanatory comments, choice of identifiers, and appropriate use of functions
- correctness of the implementation on unseen test input

It is perfectly possible to obtain a respectable mark in the examination even if the program that you submit is not complete or correct. Credit will be given, under each of the above headings, for progress made towards a solution.