Department of Computer Science

COS132 2016
Practical 2

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

1 Introduction

1.1 Submission

The submission deadline for this practical is **26 February 2016 at 19:00**. Make sure that you have submitted your code to Fitchfork by then and have posted the required message to the discussion board. **No late submissions will be accepted**.

The upload slots for submission will be available from 22 February. Early uploads may appear and vanish before 22 February. Each of these uploads allows only three tries and will close after the first 30 people have uploaded to it. These submissions will be assessed manually. If you have participated in the early uploads, the best of the early mark and the mark obtained in the actual upload will count.

1.2 Mark Distribution

Activity	**Mark**
Introductory post on the discussion board	5
Task 1: Simple	6
Task 2: Access granted	10
Task 3: Distance	10
Task 4: Currency converter	12
Task 5: Fraction program	15
Task 6: Academic status	17
Total	**75**

1.3 A Serious Warning

It is in your own interest that you, at all times, act responsible and ethically. As with any work done for the purpose of your university degree, remember that the University of Pretoria will not tolerate plagiarism. Do not copy a friend's assignment or allow a friend to copy yours. Doing so constitutes plagiarism, and apart from not gaining the experience intended, you may face disciplinary action as a result. For more information read Chapter 3 of *Tricks of the trade Volume 1* as well as the University of Pretoria's plagiarism policy on `http://www.library.up.ac.za/plagiarism/index.htm`

2 The Discussion Board [5]

The discussion board on our ClickUP module is there to help students communicate with each another and to ask questions about practicals, Linux and other course-related work.

You are not in any way allowed to share your practical work on the discussion board. You are advised to attentively read Chapter 1 of *Tricks of the Trade Volume 1* before starting to participate on the discussion board. Make sure you understand the gist of Netiquette.

> **A Side Note**
>
> We have noticed that students seem ***not*** *to grasp* that the discussion board is mainly to discuss the course material and not for solving administrative issues. Most administrative issues are already answered in the study guide and many are more of a personal nature and should be addressed to cos132queries@cs.up.ac.za

2.1 Introduce yourself on the discussion board

Login on ClickUP and go to the discussion boards. **Locate**[1] the thread named Introduction started by Vreda Pieterse in the Administration forum. Post a message introducing yourself in one or two sentences in this thread. You can also get bonus marks if you update your picture.

Unlike other relevant posts to the discussion board, this introductory post does not contribute to your general discussion board participation mark, but only to the mark for this practical assignment.

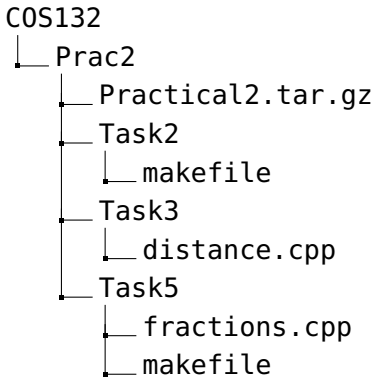
3 Download and extract

The instructions how to perform these tasks are explained in Chapter 5 of *Tricks of the trade Volume 1*

1. If you do not already have a sub-directory called `COS132`, create such directory.
2. Create a subdirectory called `Prac2` in the `COS132` directory.
3. Download `Practical2.tar.gz` from the CS website and save it in the `COS132/Prac2` directory.
4. Extract the content of the `Practical2.tar.gz` archive.

[1] Locate means to find — it is there, you have to GO to it

5. After extracting this archive your `COS132/Prac2` directory should contain the files and directories shown in the following hierarchical structure.



6. Create subdirectories for tasks 1, 4 and 6.

4 Programming tasks

The instructions how to create, compile and run programs is explained in Chapter 6 of *Tricks of the Trade Volume 1*. For each task you have to apply the programming process explained in the introduction of this chapter. You should know how to compile a single C++ file (page 64) and how to write the makefile entry to compile and link a single file (page 71).

You will notice that each of our sample makefiles contains a custom command called `run`. This is a requirement for fitchfork to be able to execute the program you have written. Custom commands are explained on pages 76-78.

Fitchfork marks by comparing the output of your program with specified expected output. For this reason you should pay close attention to the instructions for the output of your program. Also remember that names of files are case sensitive.

5 Programming task 1: Simple [6]

Write a program and save it in a file called `simple.cpp`. The program must produce the following output on three separate lines where `########` represent the eight digits in your student number. The lines of dashes should be of equal length and should contain at least 10 dashes.

```
--------------------------------------------------------------------------------
This is COS132.
My student number is u########.
--------------------------------------------------------------------------------
```

Create a tarball containing the file called `simple.cpp` and upload it using the active fitchfork assignment called **Prac2 - Simple**.

6 Programming task 2: Access granted [10]

You are given a `makefile` that can be used to compile and run a program saved in a file called `granted.cpp`.

Write a program that complies with the following specification:

1. Greet the user with the salutation **Goodday!!!**
2. Prompt the user to provide the following:
    - his/her name (a string)
    - his/her age (an integer)
3. Respond to the user's input by granting him/her access.
4. The access message should contain the name and age that was entered.

The following is a sample test run of the program (user input is shown in bold):

```
Goodday!!!
What is your name? Babajide
What is your age? 25
Access is granted to Babajide (aged 25).
```

Test the program by typing the following command:

```
make run
```

Create a tarball containing the file called `granted.cpp` and the given makefile.

Upload the tarball using the active fitchfork assignment called **Prac2 - Access granted**.

7 Programming task 3: Distance [10]

A specific underwater drone moves at a constant speed of 120 km/h. You are given a skeleton of a program called `distance.cpp`. The program should calculate the distance this drone has traveled if the time it traveled is known. Complete the program.

The following is a sample test run of the program (user input is shown in bold):

```
Enter the hours traveled: 2
The drone traveled 240 km in 2 hours.
```

Create a `makefile` that can be used to compile and run this program.

Run the program several times using different input values to verify that it works correctly. When satisfied, upload a tarball that contains the completed program and your makefile to the appropriate upload slot on the cs website.

8 Programming task 4: Currency converter [12]

When South Africans travel to the US or Europe we need to know the value of our money in the foreign countries. The current exchange rates are as follows:

- each Rand is worth 0.05917 Dollars
- each Rand is worth 0.05681 Euros

Write a program that uses these exchange rates to convert any given amount in Rand to Dollars and to Euros accurate to two decimals. The following is a sample test run of the program (user input is shown in bold):

```
Enter Rand amount: 1000
1000 Rands(s) = 59.17 Dollar(s)
1000 Rands(s) = 56.81 Euro(s)
```

Run the program several times using different input values to verify that it works correctly. When satisfied, create a tarball that contains your program and a makefile that can be used to compile it. The makefile should also contain a custom run command. Upload your tarball to the appropriate upload slot on the cs website.

9 Programming task 5: Fraction program [15]

You are given the the code of a program called `fraction.cpp` and a `makefile`. This program is written to calculate the sum of two fractions entered by the user. The user has to enter the numerator and denominator of the fractions. The following is a sample test run of the program (user input is shown in bold):

```
This is a fraction calculator
*****************************
FIRST FRACTION
  Enter the numerator: 1
  Enter the denominator: 2
SECOND FRACTION
  Enter the numerator: 1
  Enter the denominator: 2
1/2 + 1/2 = 1.00
```

The given code contains seven syntax errors, two logical errors and a two layout errors you have to fix. The layout errors are about complying to coding standards regarding user orientation as specified on page 21 of *Tricks of the trade Volume 1*.

Create a tarball containing the corrected file called `fraction.cpp` and upload it using the active fitchfork assignment called **Prac2 - Fraction program**.

10 Programming task 6: Academic status [17]

Write a program that can be used to determine the academic status of a COS 132 student when his/her final mark for the module is known. The program should prompt the user for his/her final mark and then display the academic status as defined in the following table:

Mark	Status
75 and above	Pass with distinction
50 to 74	Pass
40 to 49	Admitted to re-exam
Below 40	Fail

When the mark entered as a real value the fraction should be ignored.

The following are sample test runs of the program (user input is shown in bold):

```
Enter your mark: 88
Congratulations! Having 88 you pass with distinction.
```

```
Enter your mark: 67.8
Well done. Having 67 you pass.
```

Use your own words in the first sentence of each displayed result.

Run the program several times using different input values to verify that it works correctly. Be sure to test the program with border values such as 39, 40 and 41.

When satisfied, create a tarball that contains your program and a makefile that can be used to compile it. The makefile should also contain a custom run command. Upload your tarball to the appropriate upload slot on the cs website.