Department of Computer Science

COS132 2016
Practical 3

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Introduction

The submission deadline for this practical is **4 April 2016 at 07:30 AM**. Note that this is at the end of the recess. You should aim to complete this assignment before the recess because staff support is not available during the recess.

Fitchfork marks by comparing the output of your program with specified expected output on a line by line basis. 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.

A Serious Warning

It is in your own interest that you, at all times, act responsible and ethically.

Download and extract

1. If you do not already have a sub-directory called `COS132`, create such directory.
2. Create a subdirectory called `Prac3` in the `COS132` directory.
3. Download `Prac3_v6.tar.gz` from the CS website and save it in the `COS132/Prac3` directory.
4. Extract the content of the `Prac3_v6.tar.gz` archive.
5. After extracting this archive, your `COS132/Prac3` directory should contain the files and directories shown in the following hierarchical structure.

```
COS132
└── Prac3
    ├── Prac3_v2.tar.gz
    ├── Task1
    │   ├── findErrors.cpp
    │   └── makefile
    ├── Task5
    │   ├── countriesLarge.txt
    │   ├── countriesSmall.txt
    │   └── makefile
    └── Task6
        ├── makefileJob
        ├── makefileTest
        └── paintTools.h
```

Task 1: Finding Errors

You are given a program called `findErrors.cpp` and a makefile that can be used to compile and run this file. This program created to calculate the quotient of two given integers. The given code contains three syntax errors and three logical errors and one HCI layout errors you have to fix.

Make a copy of the given source code. Name your copy `errorsDetected.cpp`. Change the makefile so that it will compile and run this copy and then debug the program. The following is a sample test run of the program after the syntax and logical errors in the program has been removed the (user input is shown in bold):

```
Enter the numerator10
Enter the denominator: 2
The quotient of 10 divided by 2 is 5
```

Note that the program that produced this test run still contains the HCI layout error. It is about complying to coding standards regarding user orientation as specified on page 21 of *Tricks of the trade Volume 1*. You have to fix it.

Create a tarball containing only the file called `errorsDetected.cpp` in which all errors have been fixed and upload it using the active fitchfork assignment called **Prac 3 - Finding errors**.

Task 2: Maths Tutor

Write a program that can be used as math tutor for a young student to learn how to add two-digit numbers.

Start by requesting the user to input a seed value for the random number generator. Read in this value and use it as a seed in the `srand()` function. If the user specifies 0 as seed, use `time(0)` as seed to make the number truly random.

Next ask the user to specify how many questions should be asked.

For each question, the program should display two random numbers that the student should add, and wait for the student to enter the answer. If the answer is correct, the message `Your answer is correct` should be printed. If the answer is incorrect, a message should be printed showing the incorrect value as well as the correct answer.

After all questions were answered, draw a line of dashes and give the score as the number of correct answers out of the number of questions.

The following is a sample test run of the program. User input is shown in bold:

```
Give a seed value: 78
Number of questions: 2
1. 34 + 50 = 111
No. Your answer is 111. The correct answer is 84
2. 17 + 51 =  68
Yes. Your answer is correct
----------------------------------------
You scored 1 / 2
```

Create a tarball containing the file called `MathsTutor.cpp` and upload it using the active fitchfork assignment called **Prac 3 - Maths tutor**.

Task 3: Pattern Displays

Write a program that asks the user to enter a non-negative value less than 20 and then uses loops to display two triangles as shown in the following test run of the program. The number of lines in the triangle should be the number entered by the user. You may assume that the user will always enter valid values. Test your program with extreme values.

```
Enter the size: 10
--------------------------------------------------------------------------------
Pattern A
--------------------------------------------------------------------------------
+
++
+++
++++
+++++
++++++
+++++++
++++++++
+++++++++
++++++++++
--------------------------------------------------------------------------------
Pattern B
--------------------------------------------------------------------------------
++++++++++
+++++++++
++++++++
+++++++
++++++
+++++
++++
+++
++
+
--------------------------------------------------------------------------------
```

1. Create a makefile that can be used to compile and run your program.
2. Create a compressed tar archive that contains the source code of your program and its makefile. Give the archive a name such as `prac3Task4.tar.gz`.
3. Upload your TAR archive to the active fitchfork assignment called **Prac 3 - Pattern displays**. on the CS website.

Task 4: Palindrome

A number is called a palindrome if it remains unchanged when its digits are unchanged after being reversed. For instance 101 is is a palindrome because reversing it won't change it. It will still remain 101 where 102 is not a palindrome because after reversing it, it becomes 201.

Write a program to accept a non-zero positive integer number. If an invalid value is entered, display an appropriate error message and stop. If a valid number is entered, generate the reversed number. Display the entered value and its reverse. On the next line display a message whether the entered number is a palindrome or not. The message should start with the keyword **yes** if it is a palindrome and should start with the keyword **no** if it is not a palindrome.

The following are two test runs of a correct program:

```
Enter a nonzero positive number: 66766
The reverse of 66766 is 66766
Yes! This number is a palindrome.
```

```
Enter a nonzero positive number: 12345
The reverse of 12345 is 54321
No. This number is not a palindrome.
```

Create a tarball containing the file called `Palindrome.cpp` and upload it using the active fitchfork assignment called **Prac 3 - Palindrome**.

Task 5: Smallest country

Write a program called **`country.cpp`** to read a text file that contains the names and sizes of countries. The program should prompt the user to specify the file name that contains the data. If the file opens successfully, the data should be displayed on the screen in the order that it is read from the file. After reading all entries, the program must display the name of the smallest country on the list. Use the given makefile to compile and run your program.

You are given two sample input files you may use to test your program. The following hold:

- The number of items on a file is unknown.
- Each line on the file contains a name, followed by a space and a size.
- The country names to not contain spaces or punctuation.
- The value representing the size is the area of the country in km^2.
- The sizes are given as real numbers, but should be displayed as integers (rounded).

The output of your program should comply with the following specifications:

- The user should enter the file name on the same line as the prompt.
- Put a line of dashes directly under the prompt – even if the input is invalid.
- Leave no open lines.
- The entries should be numbered 1, 2, 3, . . . , right justified in 3 spaces.
- Leave two spaces after the number and then display the country name left justified in 30 spaces.
- Following the name of the country, display it's size right justified in 10 spaces.
- Put a line of dashes directly under the list of countries.
- Finally display the name of the smallest country on the list using a full sentence below this line of dashes.

The following is a sample test run.

```
Give the file name: countriesSmall.txt
-----------------------------------------------------
  1  Monaco                                2
  2  Grenada                             344
  3  Malta                               316
  4  Maldives                            300
  5  Vatican                               0
  6  Nauru                                21
  7  Tuvalu                               26
  8  Liechtenstein                       160
-----------------------------------------------------
The smallest country on this list is Vatican
```

Create a tarball containing `country.cpp` and the given `makefile`. Also include the test data (i.e. the txt files) in the tarball. Upload the tarball using the active fitchfork assignment called **Prac3 - Smallest country**.

Task 6: Paint Job Estimator

Write a modular program that can be used to calculate the cost of a paint job. For every 110 square meter of wall space, one litre of paint, and eight hours of labour work are required. The company charges **R63.20** per hour for labour. To simplify the problem, the following assumptions are made:

- For a paint job everything is painted using the same paint.
- All windows have a fixed size area
- All doors have a fixed size area
- All walls are completely painted (no tiles or fixtures)

Task 6.1: Paint tools

You are given a header file called **`paintTools.h`** and a makefile called **`makefileTest`**.

You have to implement the functions in this header file in a file called **`paintTools.cpp`**. This file should contain the following pre-processor directive to include the given header file:

```
#include "paintTools.h"
```

Write your own driver to test these functions. Call the driver **`testPaintTools.cpp`**. It should also contain the above pre-processor directive. Use the following command to compile these files and run the test.

```
make run -f makefileTest
```

Create a tarball containing *only* **`paintTools.cpp`** and upload it to the active Fitchfork slot called **Prac 3 - Paint tools**.

Task 6.2: Paint job

Write a modular program that uses these functions to calculate the cost of a paint job and produce the output that is shown in the test run that follows.

The program should prompt the user for the following items in the given order. Each prompt should be on a new line.

- number of rooms
- height of the walls in the building

For each room also input the following:

- the floor size of the room (width and length).
- number of doors
- number of windows

The output of the program should display all values accurate to two decimal places. The following results should be displayed each on its own line:

- The total area to be painted.
- The number of one litre tins of paint required.
- The total hours of labour required.
- The total labour charged.
- The total cost of the paint job.

You are given a makefile called **`makefileJob`**.

You have to implement the program in a file called **`paintJob.cpp`**. It should use a pre-processor directive include the given header file. The program should use the functions you have implemented in **`paintTools.cpp`**

Use the following command to compile these files and run the program.

```
make run -f makefileJob
```

The following is a sample test run of the program. Your program should produce output in this format. Pay close attention to have the separator lines in the correct places. User input is shown in bold:

```
Welcome to the paint job calculator.
**************************************
Enter the number of rooms to be painted: 2
Enter the height of the walls to be painted: 2.8
--------------------------------------------------
Room 1:
Enter the length: 3.5
Enter the breadth: 3.5
Enter the number of doors: 1
Enter the number of windows: 0
--------------------------------------------------
Room 2:
Enter the length: 2
Enter the breadth: 4
Enter the number of doors: 2
Enter the number of windows: 3
**************************************
The area to be painted is 66.77 square meters
The number of tins of paint required is 1 tin(s)
The hours of labour required is 4.86 hours
The labour charge is R306.90
The total paint job cost is R504.40
```

Create a tarball containing *only* **`paintJob.cpp`** and upload it to the active Fitchfork slot called **Prac3 - Paint job**.