# Extra Practical

COS132

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Department of Computer Science

Deadline: 5 April 2024 at 17:30:00

Marks: 30

## 1 General instructions:

- This practical should be completed individually; no group effort is allowed.
- Be ready to upload your practical well before the deadline, as no extension will be granted.
- **The deadline is final. No further extension will be granted.**
- You may not import any libraries that have not been imported for you.
- If your code does not compile, you will be awarded a zero mark. Only the output of your program will be considered for marks, but your code may be inspected for the presence or absence of certain prescribed features.
- All submissions will be checked for plagiarism.
- Read the entire practical before you start coding.
- You will be afforded 15 upload opportunities per task.
- **You have to use C++98 in order to get marks**

## 2 Plagiarism

The Department of Computer Science considers plagiarism a serious offence. Disciplinary action will be taken against students who commit plagiarism. Plagiarism includes copying someone else's work without consent, copying a friend's work (even with permission) and copying material from the Internet. Copying will not be tolerated in this course. For a formal definition of plagiarism, the student is referred to http://www.library.up.ac.za/plagiarism/index.htm. **If you have any questions regarding this, please ask one of the lecturers to avoid misunderstanding.**

## 3 Outcomes

Upon successful completion of this practical, students will be able to:

- Crafting a basic program structure in C++.
- Utilizing the standard output stream (std::cout) to display text.
- Understanding the role of the main function as the entry point of a C++ program.
- Understand variables and their various data types.
- Use arithmetic operations and mathematical expressions.

You are advised to consult the Practical 1 specification for information on aspects of extracting and creating archives as well as compilation if you need it. Also, consult the provided material if you require additional clarity on any of the topics covered in this practical

## 4 Structure

This practical consists of 3 tasks. Each task is self-contained and all of the code you will require to complete it will be provided in the appropriate Task folder. Each task will require you to submit a separate archive to an individual upload slot. That is, each separate task will require its own answer archive upload. You will upload Task 1 to Extra Practical Task 1 and so on.

## 5 Mark Distribution

| **Activity** | **Mark** |
|---|---|
| Task 1 - Simple Output | 2 |
| Task 2 - Debugging Exercise | 18 |
| Task 3 - Staff Management System | 10 |
| **Total** | **30** |

Table 1: Mark Distribution

# 6 Resources

Here are some additional resources you can use to help you complete the practical

- Data types in C++: `https://www.w3schools.com/cpp/cpp_data_types.asp`
- Constant variables in C++: `https://www.w3schools.com/cpp/cpp_variables_constants.asp`
- Basic operators, including Arithmetic Operators: `https://www.w3schools.com/cpp/cpp_operators.asp`
- Something More Advanced - Integer Division: `https://linuxhint.com/integer-division-cpp/`

# 7 Tasks

## 7.1 Task 1

In this task, you are a manager at the bespoke "Code Brew" café. Your first task is to greet employees digitally as they sign in for the day.
Your program will include a friendly greeting including your unique student number.
You have been given a skeleton task1.cpp. Edit this file to output the greeting. Your output should match the following exactly.

```
Good morning, Code Brew team! This is your manager.
Let's make today another day of great code and great coffee!
```

Note that both sentences should be on a new line, with a new line at the end of the second line.
Pay special attention to punctuation, spaces and capitalisation. Your output should match the given output exactly.

## 7.2 Task 2

Your task is to fix and complete the code given to you in the file called task2.cpp, in the task2 folder. The code is given below for your perusal. Pay special attention to the instructions below on how to "fix" or "complete" the code: running code does not guarantee full marks. **Do not modify the cout statements or any code above line 18**

Listing 1: Task 2 - Debugging Exercise

```cpp
#include <iostream>
#include <string>

void identifyType(int) {
    std::cout << "int" << std::endl;
}
void identifyType(float) {
    std::cout << "double" << std::endl;
}
void identifyType(double) {
    std::cout << "double" << std::endl;
}
void printBreak() {
    std::cout << "###" << std::endl;
}

int main() {
    // DO NOT CHANGE ANY CODE ABOVE THIS LINE
    printBreak();

    // 1
    int x = 3.9;
    std::cout << x << std::endl;
    identifyType(x);

    printBreak(); // do not change

    // 2
    int y = 2.7;
    std::cout << y << std::endl;
    identifyType(y);

    printBreak(); // do not change

    // 3
    std::string text = 'CorrectThisString';
    std::cout << text << std::endl;

    printBreak(); // do not change

    // 4
    int subtractionResult = 5 + 3;
    std::cout << subtractionResult << std::endl;

    printBreak(); // do not change

    // 5
    int safeDivisionDenominator = 0;
    std::cout << safeDivisionDenominator << std::endl;
```

```
    int safeDivisionResult = 4 / safeDivisionDenominator++;
    std::cout << safeDivisionResult << std::endl;
    std::cout << safeDivisionDenominator << std::endl;

    printBreak(); // do not change

    // 6
    double preciseDivisionNumerator = 5.0;
    int preciseDivision = preciseDivisionNumerator / 2.0;
    std::cout << preciseDivision << std::endl;
    identifyType(preciseDivision);

    printBreak(); // do not change

    // 7
    int numerator3 = 10;
    int denominator3 = 4;
    float impreciseDivisionResult = numerator3 / denominator3;
    std::cout << impreciseDivisionResult << std::endl;
    identifyType(impreciseDivisionResult);

    printBreak(); // do not change

    // 8
    double num1 = 4.4;
    double num2 = 2.5;

    std::cout << "Sum␣of␣num1␣and␣num2␣=␣" << () << std::endl; //Replace () with
       the correct arithmetic expression
    std::cout << "num1␣divided␣by␣num2␣=␣" << () << std::endl; //Replace () with
       the correct arithmetic expression
    std::cout << "The␣product␣of␣num1␣and␣num2␣=␣" << () << std::endl; //Replace
       () with the correct arithmetic expression

    printBreak(); // do not change
}
```

Listing 1: Task 2 - Debugging Exercise

**Instructions to correct the code:**

1. Variable x: Change the variable type to accurately represent its value without loss of precision (do not modify the value).
2. Variable y: Round the variable value so it aligns with the type it is being assigned to (do not modify the type).
3. Variable text: Correct the syntax error to ensure the string is appropriately assigned.
4. Subtraction Calculation: Correct the code to perform a subtraction instead of the current operation.
5. Safe Division: The aim is to avoid a division by zero and ensure safeDivisionResult equals 4 while safeDivisionDenominator starts at 0 before the division and is incremented to 1 immediately before the division occurs (use the correct unary operator). Correct the operation on safeDivisionDenominator to achieve the desired result.
6. Precise Division: Fix the type of preciseDivision to ensure the output includes decimal values.
7. Imprecise Division Result: For the division numerator3 / denominator3 to include the fractional part in the result, adjust the code to correctly perform floating-point division.
8. For each cout statement fill in the correct arithmetic expression in the ()
   (a) Sum of num1 and num2
   (b) num1 divided by num2
   (c) The product of num1 and num2

**Do not alter the printBreak() lines.**

## 7.3 Task 3

You are a software engineer tasked with developing a simple staff management system for "Code Brew" café. The goal is to digitalize their staff scheduling and payroll using C++ to efficiently manage hours worked, calculate wages, and handle bonuses and tips.

### 7.3.1 Part 1: Staff Information Initialization

NB: Do not modify any `cout` statements in the first part of this practical. Work within the designated sections of the `task3.cpp` file.

Initialize the following variables to simulate the café's staff management system:

- A string variable named `employeeOfTheMonth` storing the name of the café's most valued employee: "Jamie".
- An int variable named `hoursWorked` representing the total hours worked by Jamie last week (40 hours).
- A float variable for tracking the hourly wage rate (R20.50), named `hourlyWage`.
- A float variable named `weeklyBonus` indicating the bonus earned by Jamie last week (R200.00 for excellent customer service).
- A float variable named `totalTips` representing the total tips Jamie earned last week (R150.50).
- A boolean variable named `workedOverTime` set to true

After declaring and initializing these variables, alongside the given `cout` statements, ensure your code compiles and runs as expected. The output should match:

```
###
Employee of the Month: Jamie
###
Hours worked last week: 40
###
Hourly wage rate: R20.50
###
Weekly bonus: R200.00
###
Total tips: R150.50
###
Overtime: 1
###
```

### 7.3.2 Part 2: Payroll Calculation

For this part, you'll update the staff management figures based on daily operations.

- Jamie worked 2 extra hours on a particular day. Update `hoursWorked` accordingly.
- Calculate Jamie's total wage for the week (not including bonuses or tips) and store it in a variable called `totalWage`. Consider `hoursWorked` and `hourlyWage` for this calculation.
- Add `weeklyBonus` and `totalTips` to `totalWage` to calculate Jamie's total earnings for the week. Save this to a variable called `totalEarnings`.
- *Consider whether `totalWage` and `totalEarnings` should be integers or floats.*
- Print out the updated value of `hoursWorked` on a new line.
- Print out the value of `totalWage` on a new line.
- Print out the value of `totalEarnings` on a new line.
- Ensure there's a newline after each printed value, including `totalEarnings`.

In Part 2, only print the value when asked, with no additional text.

# 8 Submission checklist

For Task 1:

- Archive (zip) task1.cpp and rename the archive uXXXXXXXX.zip where XXXXXXX is your student number
- Upload the archive to Fitchfork Extra Practical Task 1 before the deadline
- **Ensure that if you download your code from the online compiler that you rename the file to task1.cpp before archiving and uploading**

For Task 2:

- Archive (zip) task2.cpp and rename the archive uXXXXXXXX.zip where XXXXXXX is your student number
- Upload the archive to Fitchfork Extra Practical Task 2 before the deadline
- **Ensure that if you download your code from the online compiler that you rename the file to task2.cpp before archiving and uploading**

For Task 3:

- Archive (zip) task3.cpp and rename the archive uXXXXXXXX.zip where XXXXXXX is your student number
- Upload the archive to Fitchfork Extra Practical Task 3 before the deadline
- **Ensure that if you download your code from the online compiler that you rename the file to task3.cpp before archiving and uploading**