Practical 3

COS132

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

Department of Computer Science

Official Deadline: 15/03/2024 at 17:30

Extended Deadline: 17/03/2024 at 20:00

Marks: 25

1 General instructions:

- This assignment 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 extended deadline has been put in place in case of loadshedding or other unforeseen circumstances. 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.
- **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:

1. Understand and apply boolean operators in C++ to perform logical operations.
2. Develop proficiency in using nested Boolean expressions to evaluate complex conditions, and to enhance problem solving skills in programming.
3. Demonstrate the ability to manipulate and compare variables of different data types (e.g., `int`, `char`, `double`) using boolean logic.
4. Apply short-circuit evaluation in boolean expressions to optimize code performance and avoid unnecessary computations.
5. Correctly use unary negation (`!`) to invert boolean expressions, understanding its impact on logical operations.
6. Implement boolean expressions to perform input validation and ensure that program inputs meet specified criteria.
7. Gain practical experience in C++ programming by identifying and correcting syntax and logical errors in given code snippets.

These outcomes aim to solidify foundational programming concepts in C++ related to boolean logic and operator usage.

4 Structure

This practical consists of two 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 Practical 3 Task 1 and so on. You can assume that all input will be valid (i.e. of the correct data type and within range)

5 Mark Distribution

Activity	**Mark**
Task 1 - Faulty Code	15
Task 2 - Inventory Setup	10
Total	

Table 1: Mark Distribution

6 Tasks

6.1 Task 1

Your task is to fix the code given to you in the file called task1.cpp, in the task1 folder. The code is given below for your perusal. Pay special attention to the instructions below on how to "fix" the code: running code does not guarantee full marks.

Listing 1: Task 1 - Fix the Code

```
#include <iostream>
#include <string>
void printSection() {
    std::cout << "###" << std::endl;
}
//If you change anything above here - you will get 0 for the practical !!!
int main() {
        // 1
        printSection(); // do not change
        int a;
        std::cout<<"Enter a value for a: ";
        std::cin>>a;
        std::cout<<std::endl;

        // 2
        printSection(); // do not change
        int c;
        int d = 2;
        std::cout << "Enter the value for c: ";
        std::cin >> c;

        std::cout << "Is c equal to d? " << () << std::endl; //Replace () with
           the correct boolean expression
        std::cout << "Is c greater than d? " << () << std::endl; //Replace ()
           with the correct boolean expression
        std::cout << "Is c less than or equal to d? " << () << std::endl;
           //Replace () with the correct boolean expression
        std::cout << "Is c not equal to d? " << () << std::endl; //Replace ()
           with the correct boolean expression

        // 3
        printSection(); // do not change
        double e;
        double f = 2.5;
        std::cout << "Enter the value for e: ";
        std::cin >> e;

        std::cout << "e + f = " << () << std::endl; //Replace () with the
           correct arithmetic expression
        std::cout << "e - f = " << () << std::endl; //Replace () with the
           correct arithmetic expression
        std::cout << "e * f = " << () << std::endl; //Replace () with the
           correct arithmetic expression

        // 4
        printSection(); // do not change
        int g;
        std::cout << "Enter a value for g: ";
        std::cin >> g;
```

```
    bool isOdd = (); //Replace () with the correct boolean expression
    std::cout << isOdd << std::endl;

    // 5
    printSection(); // do not change
    int h;
    std::cout << "Enter a value for h between 1 and 100 inclusive: ";
    std::cin >> h;
    bool isInRange = (); ///Replace () with the correct boolean expression
    std::cout << isInRange << std::endl;

    // 6
    printSection(); // do not change
    int i,j,k;
    std::cout << "Enter a value for i: ";
    std::cin >> i;
    std::cout << "Enter a value for j: ";
    std::cin >> j;
    std::cout << "Enter a value for k: ";
    std::cin >> k;
    bool isAscending = (); //Replace () with the correct boolean expression
    std::cout << isAscending << std::endl;

    // 7
    printSection(); // do not change
    int l,m;
    std::cout << "Enter a value for l: ";
    std::cin >> l;
    std::cout << "Enter a value for m: ";
    std::cin >> m;
    bool isEitherPositive = (); //Replace () with the correct boolean
       expression
    std::cout << isEitherPositive << std::endl;

    // 8
    printSection(); // do not change
    bool n,o,p;
    std::cout << "Enter a value for n: ";
    std::cin >> n;
    std::cout << "Enter a value for o: ";
    std::cin >> o;
    std::cout << "Enter a value for p: ";
    std::cin >> p;
    bool onlyOneTrue = (); //Replace () with the correct boolean expression
    std::cout << onlyOneTrue << std::endl;

    // 9
    printSection(); // do not change
    int q;
    std::cout << "Enter a value for q: ";
    std::cin >> q;
```

```
        bool isPositiveAndEvenOrMultipleOf5 = (); //Replace () with the correct boolean expression
        std::cout << isPositiveAndEvenOrMultipleOf5 << std::endl;

        // 10
        printSection(); // do not change
        int r;
        std::cout << "Enter a value for r: ";
        std::cin >> r;
        int s;
        std::cout << "Enter a value for s: ";
        std::cin >> s;
        bool isPositiveSumNotNegativeDifference = (); //Replace () with the correct boolean expression
        std::cout << isPositiveSumNotNegativeDifference  << std::endl;
        printSection(); // do not change
}
```

Listing 1: Task 1 - Fix the Code

Instructions to fix faulty code:

1. After inputting variable a, print **a** out in the next cout.
2. Give the correct boolean operations for each cout statement in the ()
   (a) Check if c is equal to d
   (b) Check if c is greater than d
   (c) Check if c is less than or equal to d
   (d) Check if c is not equal to d
3. Give all the correct mathematical operators for each cout statement in the ()
   (a) Calculate the sum of e and f
   (b) Calculate the difference between e and f
   (c) Calculate the product of e and f
4. Determine if the value for **g** entered by the user is odd by completing the expression assigned to `isOdd` (Hint: Look at the `%` operator)
5. Determine if the value for **h** entered by the user is between 1 and 100 inclusive by completing the expression assigned to `isInRange`
6. Determine if **i**, **j** and **k** entered by the user are strictly ascending in value (thus no two numbers can be equal). Complete the boolean expression assigned to `isAscending`.
7. Determine if either **l** or **m** or both are positive (0 is not positive). Complete the boolean expression assigned to `isEitherPositive`.
8. Determine if **exactly one** of **n**, **o**, **p** is true. Complete the boolean expression assigned to `onlyOneTrue`.
9. Determine if **q** is positive and either even or a multiple of 5. Complete the boolean expression assigned to `isPositiveAndEvenOrMultipleOf5`.
10. Check if the Sum of **r** and **s** is Positive and Their Difference (**r** - **s**) is Not Negative. Complete the boolean expression assigned to `isPositiveSumNotNegativeDifference`.

6.2 Task 2

After you made the inventory system for "Code Brew", the business grew and expanded with a more complex menu so they are asking for your help again. The cafe wants you to track their new menu items and manage the ordering of new stock in C++.

6.2.1 Part 1: Setting Up the Inventory

Inventory Initialization: Initialize the following variables to simulate the café's inventory system.

- You must assign the cafe's best selling item ("Double Nitro Mocha") into a string variable called `bestSellingItem`.
- Two variables for tracking the number of coffee beans needed for one "Double Nitro Mocha" (10) and milk in litres for the "Double Nitro Mocha" (0.5) called `beansNeeded` and `milkNeeded`.
- The barista needs to add coffee beans and milk to the machine. The input statements are given at the start of the main function and stored in `coffeeBeansAdded` and `milkAdded` respectively. Do not change these
  - Declare a boolean variable called `moreBeans` that stores whether the barista added more beans than needed. You should use `coffeeBeansAdded` and `beansNeeded` with an appropriate boolean operator to calculate this.
  - Declare a boolean variable called `lessBeans` that stores whether the barista added fewer beans than needed. You should use `coffeeBeansAdded` and `beansNeeded` with an appropriate boolean operator to calculate this.
  - Declare a boolean variable called `equalMilk` that stores whether the barista added the exact amount of milk as needed. You should use `milkAdded` and `milkNeeded` with an appropriate boolean operator to calculate this.
- Calculate the total cost of a "Double Nitro Mocha" if 1 bean costs R0.375 and 0.1l of milk costs R2.45. Store the answer into a variable called `totalCost` and use the variables `beansNeeded` and `milkNeeded`.

Once you have declared and initialised the variables, run your code along with the given cout statements to make sure that everything is working. Do not make changes to the cout statements. For the input 11 beans and 0.6 l milk the output should be:

```
How many beans have you added? 11
How much milk have you added? 0.6
###
Best-selling item: Double Nitro Mocha
Number of coffee beans an order takes: 10
Amount of milk, in litres, an order takes: 0.5l
More beans: 1
Less beans: 0
```

```
Equal milk: 0
Cost of one order: R16
```

7 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 Practical 3 Task 1 before the deadline
- **Ensure that if you download your code from the online compiler 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 Practical 3 Task 2 before the deadline
- **Ensure that if you download your code from the online compiler you rename the file to task2.cpp before archiving and uploading**