Software Development 2 Coursework

The coursework meets the following module learning outcomes:

- Use types and type-error messages to write and debug programs.
- Convert data from one format to another.
- Trace the execution of a variety of code segments and write summaries of their computations.
- Classify common input validation errors and write correct input validation code.
- Demonstrate the identification and graceful handling of error conditions.

In this coursework you are tasked with developing a system by using C++ program language for a given task. You are expected to develop the task individually and meet all the requirements been given.

This coursework is worth 35% of the module total. The submission deadline for this coursework is 30th April 23:59.

Task Description

In this section, you are given a scenario presented the system requirements. You are expected to follow the scenario to design and develop the system accordingly.

SCENARIO:

You have been employed by ASD Supermarket to design their checkout system. This system you develop is expected to collect the particulars of shoppers and the items they bought in order to calculate the cost. The following are the expected inputs:

- The name of the shopper
- The address of the shopper
- The post code of the shopper
- The payment card number
- The expiry dates.
- The secret code.

The unit for the following items:

- Baked beans
- Popcorn
- Evaporated Milk
- Bread

Note: **The unit price** for these items is stated below:

- Baked beans: £1.20
- Popcorn: £0.80
- Evaporated Milk: £ 1.15
- Bread: £ 2. 34

TASKS:

- All the outputs must be converted to capital letter.
- All input must be effectively validated.
- There is no symbol in the name such as @, £, $, % and others.
- The date must be in the format DD/MM/YY
- The unit of each item must be multiplied with the unit price to get the cost of each item.
- The total cost must be calculated.
- VAT of 20% must be calculated for the total cost.
- The value of VAT must be added to the total cost and included in the output.
- The output must be in the receipt format.

Requirements and Submission

The following requirements cover all parts of the coursework.

Your application must be in C++ using the standard library This means:

- **You cannot use libraries that are not provided by standard C++.**
- **Your code must compile using Microsoft's standard C/C++ compiler, which is part of Visual Studio.** If your code is specific for macOS or Linux then it will be marked down accordingly.

Your code must be submitted in Moodle by the deadline provided. The submission must be your own work. **If it is suspected that your submission is not your own work then your work will be referred for an academic misconduct investigation.**

The submission to Moodle must be a **ZIP file** containing the following:

- The code file(s) required to build your application.
- A read me file indicating the difficulties you met and your solutions (bugs you've fixed and/or the features implemented).

Your ZIP file **must** use your student ID as a name. For example, if your student ID is abc1234, your ZIP file should be called `abc1234.zip`. **All coursework submissions must be demonstrated to the module tutor in class during lab time. Any submissions not demonstrated to the module tutor will be marked at zero.**

Rubric

Criteria	Excellent	Very Good	Good	Developing	Not Attempted
Solution the completeness of the code to meet the specification given.	A completed solution meeting all the specifications.	A completed solution that does not quite meet all the specifications.	A completed solution that meets approximately 50% of the specification.	A completed solution is implemented and runs but lacks much of the specification.	The solution doesn't run or does not meet the specifications defined.
Correctness ability to create code that reliably produces correct answers or appropriate results.	The program produces correct answers or appropriate results for all inputs tested.	The program produces correct answers or appropriate results for most inputs.	The program approaches correct answers or appropriate results for most inputs but can contain miscalculations.	The program often produces incorrect answers or inappropriate results, but is almost there in many cases.	The program does not produce correct answers or appropriate results for most inputs.

Criteria	Excellent	Very Good	Good	Developing	Not Attempted
Logic ability to use correct program structures appropriate to the problem domain.	Program logic is correct with no known boundary errors, and no redundant or contradictory conditions.	Program logic is mostly correct but may contain an occasional boundary error or redundant or contradictory condition.	Program logic is generally correct but contains a few boundary errors and redundant conditions.	Program logic is on the right track but shows no recognition of boundary conditions (such as < vs. <=).	The program contains some conditions that specify the opposite of what is required (less than vs. greater than), confuse Boolean AND/OR operators, or lead to infinite loops.
Robustness ability of the program to handle unexpected input and error conditions correctly as evidenced via testing.	The program handles erroneous or unexpected input gracefully; action is taken without surprising the user. Boundary cases are considered and tested.	All obvious error conditions are checked for and appropriate action is taken. Nearly all boundary cases are considered and tested.	Some obvious error conditions are checked for and some sort of action is taken. Most boundary cases are considered and tested.	The program deals with the most obvious error conditions, but the program will often just exit or restart when this occurs.	The program often fails or fails completely. Boundary conditions are not tested for.

Criteria	Excellent	Very Good	Good	Developing	Not Attempted
Clarity ability to format and document code for human consumption (Good Style)	Program contains appropriate documentation for all major functions, variables, or non-trivial algorithms. Formatting, indentation, and other white space aids readability.	The program contains some documentation on major functions, variables, or nontrivial algorithms. Indentation and other formatting is appropriate.	The program contains some key documentation such as in functions and variables, but the documentation is not descriptive or helpful.	The program contains some documentation (at least the student's name and the program's purpose) but has occasionally misleading indentation.	The program contains no documentation, or grossly misleading indentation.
Submission. The submission meets norms described for the coursework.	Coursework has been submitted in a correct zip file, including a read me, and contains all the files necessary to run the program.	Coursework submission missing a particular factor (e.g., incorrectly named zip, not a ZIP, or read me missing).	Coursework submission two or more factors, or maybe missing files that are necessary for building.	Coursework submission is missing several factors.	Coursework submission missing many factors and is not in a quality state for a professional submission.

Academic Misconduct

All submissions will be processed through a code plagiarism tool. If signs of misconduct are found, all students involved will be contacted to discuss further steps. Please see here for information on academic integrity at the university https://portal.roehampton.ac.uk/information/Pages/Academic-Integrity.aspx.

Our guiding principle is that academic integrity and honesty are fundamental to the academic work you produce at the University of Roehampton. You are expected to complete coursework that is your own and which is referenced appropriately. The university has in place measures to detect academic dishonesty in all its forms. If you are found to be cheating or attempting to gain an unfair advantage over other students in any way, this is considered academic misconduct and you will be penalised accordingly. Please don't do it.