COSC2138/CPT220: Programming Principles 2A
Study Period 2 2015
General Assignment Information

This unit has as one of its requirements the completion of a practical work component worth 35% of the overall assessment. This practical work component forms one of 3 major hurdles, the others being a 5 Weblearn tests worth 5% and the final examination worth 60%. The 35% weighting translates into hands-on programming work for both assignments. The broad learning objectives are:

- To gain proficiency in C programming to the extent that students gain familiarity with the most commonly used and moderately complex features of the language.
- To develop a good understanding and competency of programming techniques such as coding style, presentation, modularity, source code management, and debugging.
- To relate algorithmic design with code implementation and important factors in good software development practice.

In line with such objectives there are two distinct assignments, so that different elements of software development such as programming in C and overall design and implementation are exercised incrementally. Assignment #1 is worth 15% and assignment #2 is worth 20%

General Requirements

These are general requirements that are applicable for both assignments.

General Requirement #1 – Buffer Handling

This requirement relates to how well your programs handle “extra input”.

To aid you with this requirement, we want you to use the *readRestOfLine()* function provided in the start-up code. Use of this function with *fgets()* is demonstrated on Blackboard in the *Sample Code*, in the *Input Validation Examples* section.

Marks will be deducted for the following buffer-related issues:

- Prompts being printed multiple times or your program "skipping" over prompts because left over input was accepted automatically. These problems are a symptom of not using buffer overflow code often enough.
- Program halting at unexpected times and waiting for the user to hit enter a second time. Calling your buffer clearing code after every input routine without first checking if buffer overflow has occurred causes this.
- Using **fflush(stdin)**. This function is not appropriate for reasons mentioned in the course FAQ on Blackboard: “Course Documents” -> “FAQ” -> “Alternative to non-standard fflush(stdin);”
- Using **rewind(stdin)**. This function is not appropriate as it is not intended to be used for buffer clearing.
- The use of buffer clearing code may have been avoided with **gets()** or **scanf()** for scanning string input. These functions are not safe because they do not check for the amount of input being accepted. Do not use either function in your assignments
- Using long characters arrays as a sole method of handling buffer overflow. We want you to use the **readRestOfLine()** function.
- Other buffer related problems.

For example, what happens in your program when you try to enter a string of 40 characters when the limit is 20 characters?

General Requirement #2 – Input Validation

For functionality that we ask you to implement that relies upon the user entering input, you will need to check the length, range and type of all inputs where applicable.

For example, you will be expected to check the length of strings (e.g: 1-20 characters), the ranges of numeric inputs (e.g: an integer with value 1-7) and the type of data (e.g: Is this input numeric?).

For any detected invalid inputs, you are asked to re-prompt for this input –do not truncate extra data or abort the function.

General Requirement #3 – Coding conventions/practices

Marks are awarded for good coding conventions/practices such as:

- Avoiding global variables.
- Avoiding goto statements.
- Consistent use of spaces or tabs for indentation. We recommend 3 spaces for every level of indentation. Be careful to not mix tabs and spaces. Each "block" of code should be indented one level.
- Keeping line lengths of code to a reasonable maximum such that they fit in the default "xterm" screen width. – 80 columns
- Commenting (including function header comments).
- Complete author identification on all files.
- Appropriate identifier names.
- Avoiding magic numbers.
- Avoiding platform specific code with system().
- Checking the return values of important functions such as fopen(),fgets(), malloc() and so on.
- General code readability.

Penalties

Marks will be deducted for the following:

- Compile errors and warnings.
- Fatal run-time errors such as segmentation faults, bus errors, infinite loops, etc.
- Missing files (affected components get zero marks).
- Files incorrectly named, or whole directories submitted.
- Not using start-up code.
- Changing the start-up code- you are free to add functions but you cannot alter the original functions

Programs with compile errors that cannot be easily corrected by the marker will result in a maximum possible score of 40% of the total available marks for the assignment.

Any sections of the assignment that cause the program to terminate unexpectedly (i.e: segmentation fault, bus error, infinite loop, etc) will result in a maximum possible score of 40% of the total available marks for those sections. Any sections that cannot be tested as a result of problems in other sections will also receive a maximum possible score of 40%.

It is not possible to resubmit your assignment after the deadline due to mistakes.

General Submission Information

The “Weblearn” facility is used for submission of all assignments. Assignments are not accepted by any other means (including email).

You will be informed via the Blackboard discussion board when Weblearn is enabled.

With Weblearn, it is important to note that any new submission completely overwrites the previous one. No special consideration will be given for people who submit incorrectly.

Requests for extension must be directed to the lecturer and must be substantiated with appropriate documentary evidence.

- For extensions of 7 days or less, please contact ouacsit@rmit.edu.au
- For extensions over 7 days, or continuing special consideration, please contact the Special Consideration Unit: http://www.rmit.edu.au/students/specialconsideration

When/how do I get my marks?

Assignment marks will be made available to students via email, approximately two weeks after the final submission deadline. An announcement will be made on the Blackboard discussion board when marks are available. An announcement will also be made with regards to what you need to do if there are any mistakes in your marks.

Help!

Please utilise the following with regards to getting help for your assignments:

- For general assignment questions, please use the Blackboard discussion board. That way all students can see all questions once.
- Please do not post large pieces of code to the Blackboard discussion board for plagiarism reasons. Email your instructor (cpt220-instructor@cs.rmit.edu.au) instead.
- If you are having problems that may prevent you from completing your assignment on time, please talk to the instructor as early as possible.

If you find any problems with the assignment specifications, please post your queries to the Blackboard discussion board.