COMP1038 Coursework 2 – An airstrike planner game

Introduction

This is the second COMP1038 Coursework. It is worth **45% of the module mark**. It requires you to write an interactive, menu-driven program that acts as an airstrike game and a report. The deadline for this exercise is **23:00 on Friday 20th of December 2024**.

Read the entire document before beginning the exercise.

If you have any questions about this exercise, please ask in the Q&A forum on Moodle, after a tutorial session, or during the advertised office hours. Do not post your program or parts of your program to Moodle as you are not allowed to share your coursework programs with other students. If any questions require this exercise to be clarified, then this document will be updated and everyone will be notified via Moodle.

Submission

You must submit **two files**: a C source code file containing all your code for this exercise and a report in PDF format describing the details of your program. The C code file must be called **airstrikeplanner.c** and must not require any other files outside of the standard C headers which are always available. The first line of the file should be a comment which contains your student ID number, username, and full name, of the form:
// 6512345 zy12345 Joe Blogs

The file must compile without warnings or errors when I use the command
gcc -std=c99 -lm airstrikeplanner.c -o airstrikeplanner
This command will be run on our Linux server cs-linux. If it does not compile, for any reason, then you will lose all the marks for testing (common reasons in the past have been submitting a file with the wrong filename or developing your solution on your personal computer without having tested it on our Linux server). If the file compiles but has warnings, then you will lose some marks for not correcting the warnings.

The report must be named **report.pdf** and should not exceed 500 words.

Before you submit your source code file, you must complete the **Coursework submission coversheet** on Moodle.

The completed source code file should be uploaded to the Coursework 2 Submission link on the COMP1038 Moodle page. You may submit as many times as you wish before the deadline (the last submission before the deadline will be used). After the deadline has passed, if you have already submitted your exercise then you will not be able to submit again. If you have not already submitted, then you will be allowed to submit **once**.

Late submissions: Late submissions will lose 5 percentage points **per hour**, rounded up to the next whole hour. This is to better represent the large benefit a small amount of extra time can give at the end of a programming exercise. No late submissions will be accepted more than 24 hours after the exercise deadline. If you have extenuating circumstances, you should file them before the deadline.

Plagiarism

You should complete this coursework on your own. Anyone suspected of plagiarism will be investigated and punished in accordance with the university policy on plagiarism (see your student handbook and the University Quality Manual). This may include a mark of zero for this coursework.

You should write the source code required for this assignment yourself. If you use code from other sources (books, web pages, etc), you should use comments to acknowledge this (and marks will be heavily adjusted down accordingly). *The only exception to this is the example programs given in lectures and tutorials; you may use them, with or without modification, without penalty.*

You must not copy or share source code with other students. You must not work together on your solution. You can informally talk about higher-level ideas but not to a level of detail that would allow you all to create the same source code.

Remember, it is quite easy for experienced lecturers to spot plagiarism in source code. If you are having problems, you should ask questions rather than plagiarize. If you are not able to complete the exercise, then you should still submit your incomplete program as that will still get you some of the marks for the parts you have done (but make sure your incomplete solution compiles and partially runs!).

Task

An airstrike planner game, developed by a computer science student, is available on Moodle. The game includes functionalities such as reading target files, searching for targets, planning airstrikes, and executing airstrikes.

Your task is to,

- Test the given program,
- Implement a C program with the same functionalities,
- Write a report detailing your work, and
- Submit both your program and the report.

Marking

The marking scheme will be as follows:

1. Tests (60%):

Your program will be tested against several test cases to verify if it meets the requirements. These tests are secret, but examples include:

- Handling typical valid inputs,
- Correctly dealing with boundary values,
- Properly managing invalid inputs, and
- Maintaining a user interface identical to the given executable.

If your program does not compile, **you will lose all testing marks**.

2. Report (20%):

A report detailing your program should be submitted along with the source code file. In the report, you should explain the main components, such as data structures, functions, and how they interact, and address the following questions:

- How do you determine whether a target file is valid or invalid?
- What are the criteria for identifying valid targets?
- What is the minimum distance between targets?
- How do you test your program for memory leaks?

3. Appropriate use of language features (10%):

Your program should use the appropriate C language features in your solution. You can use any language features or techniques that you have seen in the course, or you have learned on your own, as long as they are appropriate to your solution. Examples of this might be:

- Have you broken your program down into separate functions?
- Are all your function arguments being used?
- If your functions return values, are they being used?
- If you have complex data, are you using structures?
- Are you using loops to avoid repeating many lines of code?
- Do you use dynamic memory allocation properly, e.g. no memory leak?

4. Source code formatting (10%):

Your program should be correctly formatted and easy to understand by a competent C programmer. This includes, but is not limited to, indentation, bracketing, variable/function naming, and use of comments. See the textbook for examples of correctly formatting programs.

End Notes

This exercise is designed to test your understanding of key programming concepts. Submitting a working program with a complete report and following proper coding standards will maximize your marks. Good luck!