MURDOCH UNIVERSITY

ICT106 Fundamentals of Computer Systems

Semester 2, 2010

Assignment 1: C Programming Project (20%)

Report and Demo Due: Friday 15th October, 2010 4PM (Week 11)

(Optional) Submit an abstract on the requirements to your tutor by Week 4 for feedback

Requirements

In this assignment, you are required to design and develop an application program in C covering as much as possible all the operations and programming constructs you have learnt in the course. In addition, the program should serve specific objectives or purposes. The program should be based on gcc to ensure its portability. **Note that it is not essential for your program to include Graphic User Interface (GUI) as it can be system dependent.** Some suggestions for such program are given below:

Edutainment program: for school children teaching simple arithmetic or geometry.

Guessing Game – to guess a mystery number or object within a specified number of trials.

Encryption/Decryption Game – given an encrypted statement, can the user find out the message?

Travel Calculator program – Record the distance travelled and amount of fuel used then calculate the average consumption and carbon emission. Make some suggestions to the user on how to save fuel.

Exercise and/or Calories Counting – Relate the type of exercises, duration and level to the amount of calories burnt and compare to the calories intake. Calculate the weight gain/lost.

Project Abstract (Optional): you may consider submitting a one-page abstract for feedback by Week 4

Provide a description of the program to be developed. This should include:

- Objective of the program (What do you want to develop this program?)
- Significance (Why is it important? Any commercial value?)
- Initial design (How will you do it? Use a diagram if necessary.)
- Identification of Input, Output, Algorithm… (An overview of the program but with more details on the input and output.)
- References (Where did you get your information? Make sure that you do not take other person's work as yours!)

Your tutor will provide you with feedback on the abstract. You will be informed that whether you are on track, too much, too complex or too little. If it is too much, the project will unlikely to be completed and you have to drop some of the functions. Too complex means that the algorithm might be too difficult for your level. You might have to opt for a simpler and manageable solution. Too little means that your proposal is too trivial for the duration and the expected technical skills. You'll be advised to increase the complexity or functionality of the program. You should note that HD will only be granted to submissions which have exceptional performance indicating complete and comprehensive understanding of the subject; genuine mastery of relevant skills; demonstration of extremely high level of ability and initiative; and achievement of all major and minor objectives of the unit.

Final Report and demonstration of the Assignment 20% (Due date Week 11, Friday 15th October, 2010)

You have exactly NINE weeks to work on the program. You should continue with the development based on the initial design and adopt an incremental programming (IP) and modular approach to carry out the development. Your project should also base on programs or exercises from the practical sessions. However, you should consider carefully how the functions or modules are linked to one another. Make sure that you'll manage the time effectively and do not expect to complete the program in just a few days before the deadline. The following sections will give further information concerning the project.

Essential Requirements: The report should be comprehensive and the program should be well-structured. The program should have a reasonable set of functions in addition to the main function. It should use meaningful variable names, comments and good layout. Your submission is required to demonstrate good documentation and programming style.

Internal documentation (in the source code):

- a beginning comment clearly stating title, author, date, file name, purpose and any assumptions or conditions on the form of input and expected output;
- other comments giving useful low-level documentation and describing the each component;
- well-formatted readable code with meaningful identifier names and blank lines between functions.

External Documentation

1. **Title:** a paragraph clearly stating title, author, date, file names, and one-line statement of purpose.
2. **Requirements/Specification**: a paragraph giving a brief account of what the program is supposed to do. State any assumptions or conditions on the form of input and expected output.
3. **User Guide**: instructions on how to compile, run and use the program **together with screen capture**. List the environment that the program was developed and the target environment where it will run.
4. **Structure/Design**: Outline the design of your program. Give a written description, use diagrams and pseudocode.
5. **Listings**: attach source code listings, i.e., a print out (hardcopy) of program source code
6. **Testing**: describe your testing strategy (the more systematic, the better) and any errors noticed. Give hardcopy (such as screen capture) of results of testing.
7. **Limitations:** Describe program shortfalls (if any), eg, the features asked for but not implemented.

Marking Schemes:

Code: (50%)

- Program Code: Layout, use of comments, structure, compliance to standard
- Test Plan: layout, descriptions, comprehensiveness, passed columns
- Test Results: number of tests passed according to requirements and plan
- Demonstration to tutor (Internal students only) and answer questions satisfactorily

Report and Execution: (50%)

- Documentation as required above
- Program complexity, functionality and design
- Completeness: Correctly and fully implemented
- Excellence of design, innovation amd significance

NOTE:

You should develop and test the program in stages.

i. Write and test a simple program to read in the data input (such as options) and display the input and results correctly.
ii. Keep a copy of this working program.
iii. Edit a copy to incorporate one component (eg, functions for one option) and test that component. Make sure that it works and save a copy of the working program.
iv. If a component gives errors at any stage, you should leave that for a while to think about it. Go back to an earlier version of the program and try incorporating and testing a different component.
v. You must always keep a working version of the program with its test results so that you have something to submit.
vi. For components that do not perform as expected, submit a listing annotated with your ideas on the cause of the problem and possible solutions.
vii. You will be given marks for the work that you have done. In other words, even the program is not fully functional, your work on the design and testing will receive some marks.
viii. You may not receive full mark just because your program fulfils the requirements. You may lose marks if your work do not fulfil all the essential requirements (such as documentation and programming style.) Also, granting of HD is subject to demonstration of excellence in this work and above the average of the class.
ix. Non submission means no mark.

Late submission

Assessment submissions that are not received by the due date will be regarded as late unless an extension has been granted by the unit coordinator. Applications for extensions should be made as soon as a problem is experienced and must be accompanied by documentation, such as a doctor's certificate. Late work will attract a penalty of 10% (for that piece of assessment) per 24 hours late, up to and including 7 calendar days late. Work submitted more than 7 calendar days late will normally not be marked.

Back-up Copy: You must always keep a copy of your work. Your submission must include a completed (and signed) assignment cover sheet. An electronic copy of the assignment cover sheet is available in the Assignments folder at the unit resource site.