**Swinburne University of Technology Sarawak**
**COS10009**
**Introduction to Programming**
**Semester 1, 2019**

# Custom Program

**Due Date:** 5 PM on Friday of week 12 (31 May 2019).

**Submission Guidelines:** Your program and design report must be submitted as part of your portfolio in an A4 paper envelope with a facing sheet attached to the front of envelope. Late penalties will apply as described in the unit outline.

**Deliverables:** (X means required)

| | | |
|---|---|---|
| 1 | Facing sheet with your signature | X |
| 2 | All source files, data files, extra libraries, project file, and the executable file. | X |
| 3 | Printed design report, including user manual, description of program design and flow chart / structure chart. | X |
| 4 | Screen capture of the program output. | X |
| 5 | Print-out of the source code. | X |

**Purpose:**
Demonstrate that you can design and implement your own program using structured procedural programming tools.

**Task:**
Create a program of your own design by using Ruby as the programming language. You are expected to demonstrate your ability to apply the concepts learnt to the design and development of a complete program of your own design. Although you are given the freedom to choose the topic of your custom program, it MUST fulfil the minimum requirements stated below:

- Demonstrate the use of functional decomposition - implement the program with a number of functions and procedures.
- Demonstrate the use of arrays and records
- Demonstrate the use of structured programming (sequence, selection, and repetition)
- Demonstrate appropriate use coding conventions (indentation and comments)
- It must not use global variables, or goto.
- Make sure you can explain your code in an interview

**Here are some steps to get you started:**

- Think about what you want the program to do. Maybe write up a paragraph or two to explain it to others. Drawing a picture of what you want it to look like is also a great idea.
- Show your plans to your tutor or lecturer to get some feedback if you are unsure.
- Start thinking about the data - what records and enumerations will you need? (Tips: Start small, you can easily add to records at a later stage. Try to identify what records you will need, then add just the basic data - enough to get something working. Once that first part is working, add additional fields as they are needed)
- Get something working quickly. You want to see it running ASAP. Once it is working build it a little at a time, get one thing working then move on to the next aspect.

**Assessment Criteria:**

| **Grade** | **Custom Program / Code Quality Requirements** | **Design report Requirements** |
|---|---|---|
| **Pass** | Your program must include some form of logic that operates on the data within the program. To achieve this your program must:<br>- do more than just collect and display data to the user.<br>- be substantially your own design and implementation.<br>- compiles and executable runs<br>You must demonstrate the use of the following programming aspects:<br>- Functional decomposition with functions and procedures performing identified tasks.<br>- Parameter passing.<br>Your code must meet good programming practices:<br>- No use of goto and no global variables. | - An overview of the program's goals<br>- A structure chart illustrating how the core aspects of the program fit together. |
| **Credit** | Your program must include some form of logic that operates on the data within the program. To achieve this your program must:<br>- Meet all requirements of Pass | - Meet all requirements of Pass<br>- A description of core program functionality and how it works. |

| | | |
|---|---|---|
| | You must demonstrate the use of the following programming aspects:<br>- Meet all requirements of Pass<br>- Data organised using arrays<br>- Structured programming concepts to organise function and procedure logic.<br><br>Your code must meet good programming practices:<br>- Meet all requirements of Pass<br>- Artefacts should be named appropriately.<br>- Minimum comments to document its structure. | |
| **Distinction** | Your program must include some form of logic that operates on the data within the program. To achieve this your program must:<br>- Meet all requirements of Credit<br><br>You must demonstrate the use of the following programming aspects:<br>- Meet all requirements of Credit<br>- Data organised using arrays and records/custom data type<br><br>Your code must meet good programming practices:<br>- Meet all requirements of Credit<br>- Code should include sufficient comments to document its structure.<br>- Indentation must help document program structure. | - Meet all requirements of Credit<br>- A data dictionary that communicates the format of the records/custom data type in your program. |
| **High Distinction** | Your program must include some form of logic that operates on the data within the program. To achieve this your program must:<br>- Meet all requirements of Distinction<br>- Extra feature that goes beyond the scope of this unit | - Meet all requirements of Distinction<br>- Demonstrate good communication skills, and present a well thought out program design. It must clearly communicate the abstractions being created, and outline how |

| | | |
|---|---|---|
| | You must demonstrate the use of the following programming aspects:<br>- Meet all requirements of Distinction<br>- Demonstrate ability to select appropriate artefacts<br>- Design useful abstractions, and suitably organise an advanced level of functionality<br>Your code must meet good programming practices:<br>- Meet all requirements of Distinction<br>- Code will be extensively commented, with comments providing meaningful insights into the code being documented. | the functionality is organised. |

Evidence of plagiarism: