Assessment Notification

**Course: 11 Software Design and Development**

**Task Topic: Task 2 Project Year: 2018**

|  |
| --- |
|  |
| **Task Details:** |
| **Assessment Task No.2 Due Date: Week 6 Term 3 Weighting: 35%** |
|  |
| **Submission Instructions** |
| * Students must hand in a digital copy of their programming code. * Students must put all content in a Google Drive Folder shared with your Teacher. * The School Assessment policy will be followed for students handing in late work, or missing, assessment tasks. * All work must be submitted, despite how late it may be. |
|  |
| **Outcomes being assessed** |
| P2.2 explains the effects of historical developments on current practices  P3.1 identifies the issues relating to the use of software solutions  P6.1 describes the role of personnel involved in software development.  P1.1 describes the functions of hardware and software  P1.3 describes the interactions between the elements of a computer system  P4.1 analyses a given problem in order to generate a computer-based solution  P4.2 investigates a structured approach in the design and implementation of a software solution |
|  |
| **Task Description** |
| Overview: Students need to combine their knowledge about various coding techniques to create a solution to a given problem. You need to create an appropriate interactive user interface (GUI) for the applications. |
|  |
| **In order to undertake this task you need to** : |
| Tasks:  Part 1: Planning   * Select an algorithm method that you have been introduced to and design a solution to a problem from the given list. * You may need to create a number of algorithms that reflect the complexity of the problem and illustrate good planning techniques such as abstraction and stepwise refinement (breaking the problem down into smaller more manageable sub procedures. * You are creating an "event driven" user interface driven application, so consider how that would affect your Algorithms.   Part 2: Coding   1. Choosing a language that you are comfortable with that offers the creation of a Graphical User Interface (GUI) and code a solution to the problem.    1. Languages you could choose between include html/JavaScript, python (using TKInter or PyGame), visual basic, java or another language you may be familiar with. 2. Create a Project Log    1. Document debugging and error correction methodologies that you employ in the testing and coding phases of your program.    2. Document instances where you search for code help via the internet or other sources    3. Documentation in b and c above can take the form of a log with dates and actions you undertook – include screenshots.   Choices:   1. Problem: South Pacific Xtreme (SPX) Cinemas. SPX Cinemas requires a GUI based program that will ensure that it does not oversell a movie by selling more tickets than there are seats. Design and code a program that acts as a booking program for the theatre for movies that it shows.   This code should include:   * Allows people’s names to be reserved against a seat number, and * Allows for reservation for several sessions in a day.  You could use arrays (simple through to multi-dimensional) as a way to go about solving this problem  1. **Problem: Towers of Hanoi** https://upload.wikimedia.org/wikipedia/commons/thumb/0/07/Tower_of_Hanoi.jpeg/300px-Tower_of_Hanoi.jpeg  The **Tower of Hanoi** (also called the Tower of Brahma or Lucas' Tower and sometimes pluralized) is a mathematical game or puzzle. It consists of three rods and a number of disks of different sizes, which can slide onto any rod. The puzzle starts with the disks in a neat stack in ascending order of size on one rod, the smallest at the top, thus making a conical shape.   The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:   * 1. Only one disk can be moved at a time.   2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty rod.   3. No disk may be placed on top of a smaller disk.   With 3 disks, the puzzle can be solved in 7 moves. The minimal number of moves required to solve a Tower of Hanoi puzzle is 2n − 1, where n is the number of disks.  https://en.wikipedia.org/wiki/Tower\_of\_Hanoi  You have to **write code to implement the Towers of Hanoi game**. You must allow the user to choose the number of disks to start with. The minimum number are 3 disks. You need to count the number of attempts and compare this with the minimum possible moves (see below for algorithm).  This must be a GUI based game.   1. **Problem: Hangman** Miss Crabapple wants to teach her year 4 class how to spell correctly. Since they live in one of the southern states of the USA, there is nothin’ they like more than a good ol’ fashioned hangin’.      1. So she wants a GUI-based program that will choose a random word from a list and give the student a chance to guess the word by selecting letters that may or may not be in the word.    2. The program will indicate where the letters appear if they exist in the word. Show what letters they have guessed already and warn the student if they use an already guessed letter. The ‘score’ is tracked by calling an array which ‘draws’ a hangman scene one part at a time if an incorrect guess is made. It could look like this:     At the end the student either saves the criminal by guessing the word, or the unfortunate felon is hung and then the correct word is displayed.  The program must also allow Miss Crabapple to input a list of words from which to choose taken from the weeks spelling list.   1. **Problem: Connect-4**  You are to create a GUI based version of Connect-4. This is a 7 wide x 6 up grid-game, where players take turns putting one of 42 (21 each) coloured counters on top of the grid and the counter falls to the first available location.  The game ends when each player runs out of counters. At the end of the game, count up the number of "runs" of counters containing at least 4 of the same counters.   Related image |

|  |
| --- |
|  |
| **Task Rubrics** |
| Assessment Task 3 SDD - Marking Criteria /30   |  |  |  | | --- | --- | --- | | Question | **Item** | Mark Allocation | | Part 1 | Planning | **8-10**  Very good use of algorithms to plan the program structure  **5- 7** Good use of planning techniques to design program  **1–4** Simple and inadequate pseudocode / flowcharts  **0** Poor attempt not providing any planning | | Part 2 a) | Documentation of errors and help | **5** Exemplary coverage of debugging and error correction as well as comprehensive coverage of sources  **3-4** Good coverage of debugging and error correction as well as informative coverage of sources  **1-2** Elementary coverage of debugging techniques used and little referencing of source.  **0** Non serious attempt at documenting debugging and no indication of where help was sought from | | Part 2 b) | Coding  Task | **11 – 15** Program works as intended, well structured (using appropriate subprograms, and a well-designed User Interface.)  **8-10** Program syntax correct, logic correct and free of error – but program(s) not structured appropriately, and User Interface is not well-designed.  **7-9** Program logically correct and free of errors. But program poorly designed, and very basic User Interface  **4-6** Code mostly correct still in working order, program not structured well, and very little GUI elements used.  **1-3** Critical errors and flaws in program logic  **0** Failure to submit | |