Practical project

420-D02-SU

Introduction to structured programming

AEC Programming, Networks and Security (LEA.5F)

AEC Programming and Web Technologies (LEA.5G)

AEC Video Game Programming (LEA.CU)

**Evaluation weight:**

25% of the final grade

**Submission specifications:**

* **Due date: morning of the final exam**
* Name the program correctly: Library + *name1* + *name2* [ + *name3* ]
* The code of the program and any dependencies
* The compressed file deposited in Dropbox in the folder designated for this purpose

**Description:**

Create the old game Hangman in a console application. The game has a hardcoded bank of words it can pick from and have the user guess. As the user guesses incorrectly, it will draw the hanging man. Below the man will be the field \_ \_ \_ \_ .. to guess letters, which will fill as the user guesses. The user can enter “ans” to enter the full answer. The words are to be imported by a text file.

**Remarks:**

* The main menu have option for player to play
* If player plays, a screen is showed with a ASCII art Gallow and underneath it underscores(\_) equal to the number of letters to guess
* User is prompted for input, the user may enter a single char, or Ans: theAnswer to answer the question
* When user guesses correctly, the underscore is replaced with the guessed letter
* When user wins, the correct answer is shown, and enter goes to next question
* When user guesses incorrectly, a body part is added to the hanging man
* When enough guesses are incorrect, the round ends and they return to main menu after being shown the answer
* Incorrect letters guessed are displayed somewhere
* The words are imported by a text file

**Evaluation:**

The project will be graded according to the following criteria:

* **Exactitude of the program:**Does the program do what it is supposed to do?
* **Visual appearance:**All text displayed to the screen should be well arranged and written in proper English.
* **Input validation:**The program should properly handle errors that can occur during data input.
* **Structure of the code:**The program should be split up into functions according to the relevant needs.
* **Arrangement and clarity of the code:**The code should be properly indented, with relevant comments where needed, and it should respect programming conventions (variables in camelCase, constants in UPPER\_CASE\_SEPARATED\_BY\_UNDERSCORES). Hard-coded values should be put into constants (magic numbers).

|  |  |
| --- | --- |
| **Tasks to accomplish** | **Points** |
| Respect submission specifications | 1 |
| Program compiles and executes | 1 |
| Structure of the code (comments, etc) | 2 |
| Bank of words (min 40) | 1 |
| Menu | 1 |
| Gallow art | 1 |
| Hangman fills | 2 |
| Underscores display # of letters | 2 |
| Correct guesses fill underscores | 2 |
| Incorrect guesses are displayed | 1 |
| Incorrect guesses add to hangman | 1 |
| Enough incorrect end round | 1 |
| End round shows answer, waits for enter key | 1 |
| Ans: theAnswer to guess answer | 1 |
| Getting answer correct ends round | 1 |
| Each round is reset properly | 2 |
| All the words are imported by a text file | 4 |
| Exit the application | 1 |
| Input validations | 1 |
| Global evaluation (teamwork, effort) | 2 |
| **Total** (with bonus in parentheses) | 29 |