Computer Programming

Programming Assignment #2, 2011

Brief:

You are to write a console-based computer game, broadly equivalent to the traditional 'hangman' challenge.

Summary:

The program will use an external file containing a dictionary of words. You can use the in-built Linux file **/usr/share/dict/linux.words** or use the alternate file **/home/public/hangman.dat** which will be provided.

From this file words will be chosen randomly and presented in blanked-out form for the user to guess. You will use the perl module **Term::Screen** to control positioning of all elements on screen. This module and how to use it are documented on the relevant page of the course website. 'Nuggets' 6 and 7 explain and show how to use the module and its features.

Because of the need to repeatedly allow the user to guess letters of the word you will use a **while** loop; use of this is documented starting on page 14 of the on-line 'Getting Started' guide on the course web-site.

Elements:

The following elements need to be displayed:

1. The word to be guessed presented – initially blanked-out – with correctly guessed letters added as the game progresses.
2. The letters of the alphabet available to guess and/or the letters already guessed.
3. A progress indicator:
    - You may use ASCII art to represent 'graphically' the status of the player
    - Equally acceptable is a numeric status – *e.g.* guesses left – indicator.
    - A combination of both of the above as preferred by you.
4. Any relevant help instructions – such as how to end the game.
5. Headings for the relevant sections.
6. A web-based version of the game (http://www.webhangman.com/hangman.php) can be taken as a guideline of key sections and elements. *When looking at this however please do not forget that you are only required to provide a CLI text-only version of the game*.

Program Style:

This is a 'freestyle' project. Any formulation of elements and use of programming logic that enable the user to play the game is acceptable.

Therefore you should decide and immediately document and/or sketch how you want your program to operate and appear on screen. This then becomes your initial design section and will represent the target that you will be operating towards achieving.

Randomisation:

You will use the perl **rand()** function to generate a random number – this number can then be used to select a word from your chosen dictionary. The following sample code will print a random number between 1 and 1000 inclusive:

```
#!/usr/bin/perl
  $num = rand(1000) + 1;
  $num = int($num);
  print "> $num\n";
```

Termination:

Pressing the full-stop key will stop the current game.

Presentation:

Marks are awarded for attractive presentation both of the screen output and the source code. Code indentation is ***vital***.

Submission Mechanism:

By paper via the submission box in room 15, and by e-mail (**details to be announced**). Don't forget the 'My Own Work' form as a cover page. Ensure you have at least:

- Cover sheet ('My Own Work')
- Flow chart (Read the programming website for information).
- Source code (The perl program, printed)
- Sample data used (A screen capture of the program in operation)
- Other screen capture(s) as required
- Photograph of you using the game program with both you and the screen display clearly distinguishable
- Any other relevant supporting materials

Due Date:

TBA