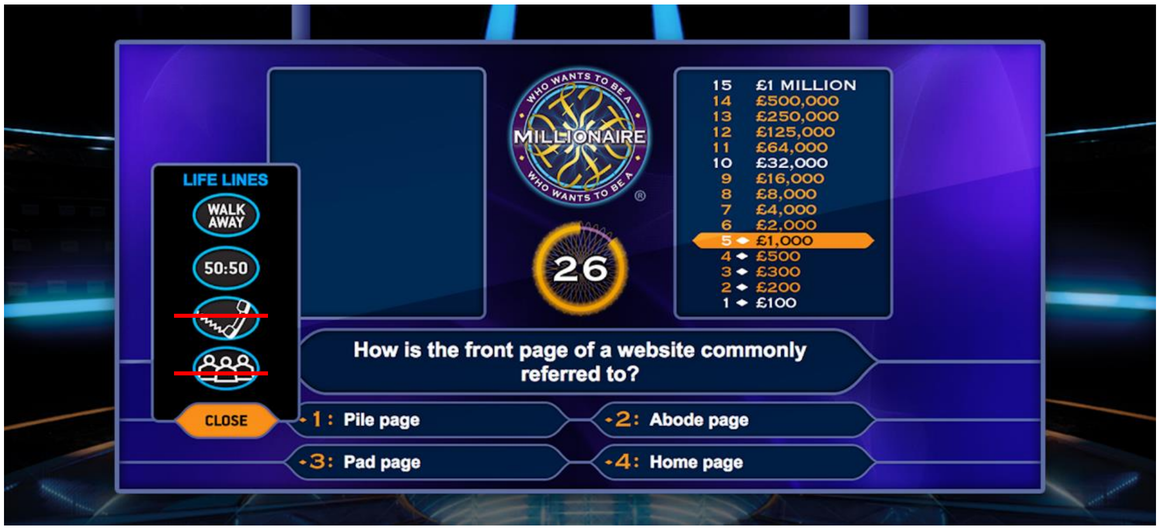
CMPS4143: Contemporary Programming Languages
(Major) Programming Assignment 4
Due: Monday, Sept. 28, 2020   75 points

PURPOSE: To develop a windows application that implements an interactive game; to use a variety of GUI components; to use a message box; to write a program using more than one class; to use arrays and array methods; to use the Random class; to use a do-while loop; and to use the ? operator. *Remember you will be graded on applying the concepts learned in class.*

PROBLEM: **"Who Wants to be a Millionaire" Game**

The Game: This game allows a contestant to discover answers to a series of questions. (Put a time limit on it for extra credit.) Contestants in your game will only have one life-line, that is the 50:50. Of course contestants can walk away with their earnings at any time in the game. If they miss a question, they lose what they have earned down to $100, $1000, $32,000.

Here is an example of the scoreboard in the game. Be creative in your look, but make it easy to understand. Make sure instructions are shown (maybe in that blank area).



Play: There are 15 rounds and three "safe havens." Round 1, 5, and 10 are safe havens. In each round, a question is revealed with four choices, A-D. The contestant enters and submits an answer. (Add a TextBox for the user to type an answer and a Submit Button.) After an answer is submitted, the game will indicate whether the answer is incorrect or correct.

If the answer is correct, the game will reveal the answer in the appropriate label and highlight the dollar amount one. If the contestant gets the answer correct, the contestant goes on to the next round. If a contestant answers all questions correctly, the contestant wins $1 million.

If an answer is incorrect – the game will be over. The contestant will win the guaranteed amount for the last safe haven, for which they answered the question correctly.

There are only two “lifelines”, which can only be used once. 50/50 – eliminates one correct and one incorrect answer from the multiple-choice selection, leaving the contestant with only one correct and one incorrect option. Use random to select the incorrect answer remaining. The other “walk-away” is the contestant opting to quit and win the amount for the last question answered. These options are available *after* a question is revealed.

Winning the game: A contestant “wins” by walking away with the dollar amount for the last question answered or when they correctly answer the million dollar questions.

DESIGN HINTS: Utilize at least two classes – a data structure and the MillionaireGame Form (which controls the main flow of the game and contains the GUI components).

Data structure class(es): It holds the questions and answers read from the input file and keeps track of the round the player is currently on, etc. Methods include:

- A constructor to perform the following tasks:
  - Initialize the data structure
  - Set the round number to zero.
- EvaluateAnswer – evaluates answer and provides the feedback reflecting whether answer is correct or not.

MillionaireGameForm -

- The form with the GUI – it controls the overflow of the game. It starts a new game, lets the user enter and submit answers, and will repeat until the user quits the program. It presents the answers of the contestant to the *DataStructure class* for evaluation and feedback.

INPUT: User should be able to

- Enter name of input file containing questions and answers. (I will supply you with file processing code – you’ll have to store in array.)
- Enter an answer.
- Indicate whether they want to use a lifeline or submit an answer for current game (may need to be enable or disable controls at appropriate times).

OUTPUT: Output is all done on a form. The form should display

- Instructions to play the game
- All the rounds (15 labels)
- Current round / dollar amount (can change background color of one of those labels)
- The current question
- Four answers
- Response to the contestant answer – indicating if correct or not.
- Button(s) to play a new game or submit an answer. I'll give you some leeway, you can use and reuse one button or display both buttons.

Upload a zip folder of:

1) Documented form and class files (.cs)
2) Screen dump(s) of image(s) when running
3) An input file you make up yourself
4) Application (.exe file). The application is in a debug folder.

EXTRA CREDIT OPTIONS:

- Incorporate images – use them on the labels to display the answer number (2 points)
- Incorporate sound effects (3 points)
- Implement a time clock – could be just a label with a counter (5 points)