Guillermo Reyes
Dr. Mark Lehr
January 30, 2015

Project 1

Who wants to be a millionaire.

Preliminary planning/Rules:

For project one, I am creating a trivia- based game similar to "Who wants to be a millionaire?". I will need to have a database that contains all the answers, questions and options. The questions will be randomly generated, and the User will only proceed if he answers the questions correctly.

1. The user will start off will zero dollars.
2. The user will be given the opportunity to answer questions
3. Each level has a retain difficulty and award to it; the harder the question, the higher the payout.

4. The game will not allow for lifelines.
5. The user will input the correct number to each corresponding choice.
6. By the default, the user will be able to make it to the final round, however, only points will be awarded if they answer the questions correctly.

About the questions:

The questions will range from common knowledge, historical events, music, and even some pop culture. The user will not choose the category type.

Format :

The game will be comprised of functions that contain the databases. For this purpose, I will be using parallel arrays that will contain the corresponding selections and answers. The whole program will run on a do-while loop.

The functions:

The functions will consist of parallel arrays, that contain the corresponding elements. In the function, a random question will be generated. Each function will have the responsibility of keeping track of the score and determining when to promote the user. Each level of difficulty will have its own function.

These functions will we declared in the main, since I need to keep track of the correct responses so I can sum up the points.

In the main function, a random number will be generated which will be inured into the function to extract the value of an array and use that information to display the question. If the answer matches the answer of the array, then the user is awarded the points or lose.

Project outline:

Databases:

The databases consists of string arrays, specifically, parallel arrays. each string has a corresponding

string that contains its options and answers.

Access to database:

To make the game more dynamic, the arrays are called randomly. When called randomly, it will display

the question, along with options. It will then verify if the answer input is correct or not.

Example code:

<enter example code of question display>

After user input, their choice will be verified as either correct or incoorect.

if incorrect, it will tell them it is incorrect and award them 0 dollars

if correct, they will be told it is correct and will be awarded with "money".

Cycle of questions:

Three questions from each difficulty will be randomly chosen, and this will go on until all five levels are

completed .

A for loop, do while, and while loop are implemented as per instructions.

Example of Loop:

<enter loop code here>

Switch statement menu()

There is a menu implemented in the code. The user will either choose to play the game, view source of

trivia, or view rules of the game.

There is a default as well as a safeguard.

PseudoCode

Display menu.

choose option/

If case a, the game will begin to run level one

generate a random number

use that random number to call arrays

verify answer.

if correct ,award money

if not correct, do not award money.

all while counter<=3;

level 2:

repeat number genaration and verifications

.

.

.

Level 5

After level 5

cout<<you have won x amount of money

prompt user to enter initials to write to a file

end game

if case b (view sourcese)

//display sources in cout statements

if case c(display rules)

//Rules will be displaye

if incorrect input

//Default statement

End program!

This program is conceptually simple, however I was having issues with string arrays and insuring that it

runs smoothly was the hardest part. I spent long nights figuring it out but.

Challenges:

The project is conceptually simple, but it was tricky optimizing the program by makingt he code neater and easier to read.