You are to create a **2-player** modified **Wheel of Fortune** game where the players will compete in guessing a mystery word or phrase in each round. At the start of the program, **load** predefined (minimum) questions and bonus questions to **linked list/s** and show the main menu: **Maintain**, **Play Game**, & **Exit**. A question is simply the set of information involving the following: **mystery word** or **phrase** (1 to 5 words, 1 to 20 characters per word), **count of words** (positive integer), **category** (at least 5 possible categories, 5 to 20 characters). A bonus question on the other hand is composed of: **question** (70 characters max), **choices** (a,b,c, and d - 20 characters max), **correct answer letter** (any letter from a to d), **category** (5 to 20 characters).

I. Maintenance

Upon choosing the **maintain** option, the user is asked to enter his **password** (5 to 12 characters) in a hidden manner and **validated**. After log-in, the user is then allowed to do any of the following:

1. **Change Password -** Ask for the new password **twice** (for confirmation) and in a hidden manner.
2. **View Questions -** Ask for a **category** of questions. The list of questions for the chosen category is shown based on the **alphabetical order of the question's mystery word**. **Divide** the questions into several pages and navigate the pages via **key press** (left/right keys, or up/down keys).
3. **View Bonus Round Questions** – Ask for a **category** of questions. The list of bonus round questions for the chosen category is shown based on the **alphabetical order of the bonus question**. **Divide** the bonus questions into several pages and navigate the pages via **key press** (left/right keys, or up/down keys).
4. **Add Questions -** Ask for the question details: **mystery word** or **phrase** (1 to 5 words, 1 to 20 characters per word), **count of words** (positive integer), **category** (5 to 20 characters). Ensure that the **mystery word** is **unique** for all categories. **Confirm** addition.
5. **Add Bonus Round Questions** – Ask for the bonus round questions details: **question** (70 characters max), **choices** (a,b,c, and d - 20 characters max), **correct answer letter** (any letter from a to d), **category** (5 to 20 characters). **Confirm** addition.
6. **Delete Questions -** Ask for a **category** of questions. Let the user **select/choose** a question. **Display** the question details and **confirm** deletion.
7. **Delete Bonus Questions -** Ask for a **category** of questions. Let the user **select/choose** a bonus question from the given category. **Display** the bonus question details and **confirm** deletion.
8. **Modify Question -** Ask for a **category** of questions. Let the user **select/choose** a question. **Display** the question details, **ask** for modifications, and **confirm** modification.
9. **Modify Bonus Question -** Ask for a **category** of questions. Let the user **select/choose** a question. **Display** the question details, **ask** for modifications, and **confirm** modification.
10. **Save Questions -** Ask the user for a filename and save all the questions in .txt format. The format is up to you. Ensure that the generated question file **can be loaded** by the program.
11. **Save Bonus Questions -** Ask the user for a filename and save all the bonus questions in .txt format. The format is up to you. Ensure that the generated question file **can be loaded** by the program.
12. **Load Questions -** Ask the user for a filename and load the questions from the file. All existing questions currently in game will be **overwritten** by the questions from the file.
13. **Load Bonus Questions -** Ask the user for a filename and load the questions from the file. All existing bonus questions currently in game will be **overwritten** by the questions from the file.

14. **Return –** Goes back to main menu.

II. Play Game

There should be **at least 3 questions and 1 bonus question** in each category to be able to play. Per game, ask for the **player names** (which should be displayed). In one **game**, there are **three rounds** (Round 1 - one word question, Round 2 – two word question, Round 3 – three to five word question). For each round, the computer **randomly** chooses the category and question as well as who will play first. The **category** must be displayed together with the **blanks** (representing the mystery word) for the question. The starting money of each player is **0** and money is **accumulated** through the three rounds. The **current money**, which is the accumulated money from all rounds of the current game should be shown. The **round money,** which is the accumulated money from the current round should also be shown.

For every turn, the player has 4 options:

a) Spin the **wheel** (done by the computer **randomizing** a value). The roulette outcome **should be based** on the game show wheel. See Figure 1

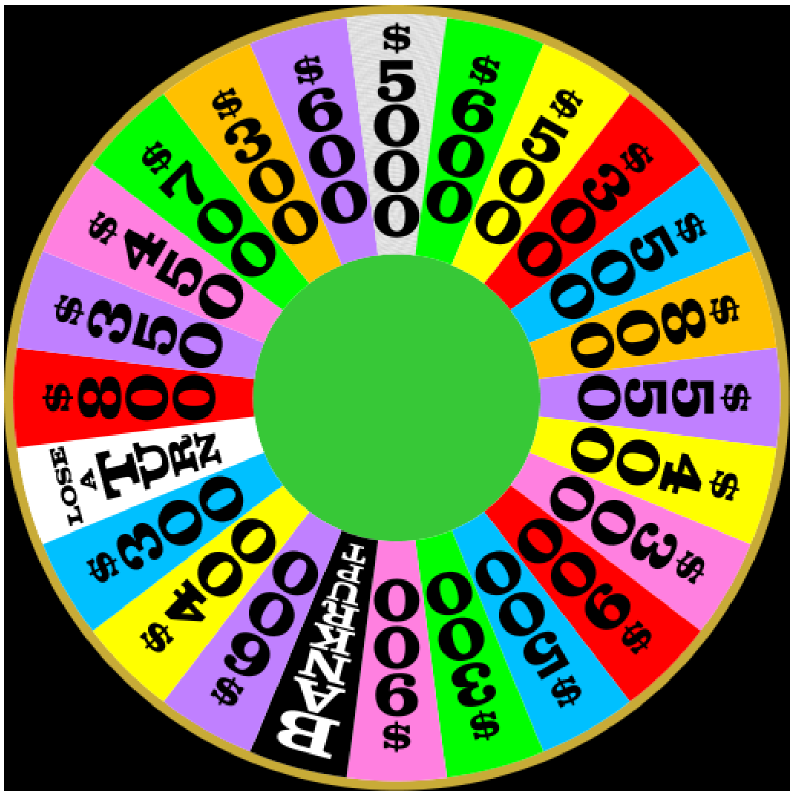

Figure 1. Game show wheel (http://classic.lagniappemobile.com/article.asp?articleID=4114)

The **probabilities** of getting a specific amount from the wheel should be implemented. There are a total of 24 segments in the wheel. So for example, the **chance of getting $600 is 12.5%**, because there are 3 segments containing $600 so 3 / 24 = 0.125. Another example, the **chance of getting $300 is 20.83%** (5 / 24). The **Lose a**

turn and Bankrupt segments **forfeit the turn** of the current player. **Bankrupt** also resets the current **round money** of the current player to 0.

After an amount is chosen, he/she must input a **consonant**.

If (a) the consonant **is in the mystery word**, the number of instances of the inputted consonant in the mystery word multiplied by the amount the player got in the wheel would be the winnings for that turn. The next turn still belongs to the current player.

Else (b) the consonant **is not in the mystery word,** the next turn is given to the other player.

Sample runs:

Sample Run 1	Sample Run 2
Mystery word: Bulbasaur Displayed: _ _ _ _ _ _ _ _ _ 1. Choose action: a.) Spin b.) Buy a vowel c.) Guess the word d.) Pass 2. Spin > randomized amount: $600 3. Chosen consonant: ‘b’ Displayed: B _ _ B _ _ _ _ _ 4. Turn winnings: $1200, computed as: 2 b’s x $600 5. Go back to step 1.	Mystery word: Bulbasaur Displayed: _ _ _ _ _ _ _ _ _ 1. Choose action: a.) Spin b.) Buy a vowel c.) Guess the word d.) Pass 2. Spin > randomized amount: $600 3. Chosen consonant: ‘t’ Displayed: _ _ _ _ _ _ _ _ _ 4. Next player gets possession of the next turn.

b) **Buy** a **vowel** – The player buys a vowel for **$250**. The player inputs a vowel. Unlike consonant multipliers, the number will not go up based on how many instances of the vowel in the mystery word. So if the player buys the letter ‘e’ and there were five Es in the mystery word, the player spends $250, not $1250.

If (a) the vowel **is in the mystery word**, no money is given to the current player. However, the next turn still belongs to the current player.

Else (b) the vowel **is not in the mystery word,** the next turn is given to the other player.

c) **Guess** the mystery word - Once a player correctly guesses the answer, he earns **$500** (**per blank** remaining in the question). The game proceeds to the next round (if there is any).

d) **Pass** – The next turn is given to the other player.

The player with the **higher amount** of money after the three rounds is declared winner. In case of a **tie**, the computer **randomly selects a winner** from the two players. The winner gets a chance to play a bonus round where he/she is given bonus multiple choice question. The **category is randomized** and a **random question** from the selected category is given. If the player answers this correctly, **he/she wins $5000**. If not, he/she retains his current money. Afterwards, the game is finished and **control** is then transferred to the **main menu**.

Quitting a game level or **quitting the game** should be **confirmed** first before executed. The player who quits is declared the loser.

All **user inputs** should be **validated**. Relevant **user prompts & messages** should be provided.

You are to design based on your creativity 4 main screens, namely **Introduction Screen**, **Maintain**, **Game Screen**, & **Closing Screen**. **Note** that **screen displays** may **vary** depending on terminal settings. Some ASCII characters maybe non-viewable in certain terminals.

III. Implementation

1. Use **GNU C compiler**. Make sure you **test** your program completely (compiling & running) in **G302A**.
2. Do not use brute force. Use **appropriate conditional** statements **properly**. Use, **wherever appropriate**, **appropriate loops** & **functions properly**. Non-use of **self-defined functions** will merit a grade of **0** for the **machine problem**.
3. You **may** use topics outside the scope of COMPRO2 but this will be **self-study**. Goto, exit, break (except in switch), global variables, are **not allowed**.
4. Follow our **coding standards** with included **internal documentation** (comments) in your program.

IV. Bare minimum requirements

Unable to meet the bare minimum requirements (**use of arrays/linked lists and functions, add questions, view questions, exit**) will merit a **0.0** for the **course**.

V. Bonus

A **maximum** of **10 points** may be given for features **over & above** the requirements. Examples of such features are:

1. top ten players / high scores
2. timer
3. Others (subject to evaluation of the teacher)

Required features must be **completed first** before bonus features are credited.

VI. Submission & Demo

Deadline: December 4/5 (W/H), class time; after class time, 0 as score for project
Requirements: Complete Program

Make sure that your implementation has considerable and proper use of arrays/dynamic lists, linked structures, files, and user-defined functions

Place the following in the CD:

- source code (Filename: <Surname_Firstname>.c)
- Executable
- function specifications* (Filename: < Surname_Firstname>_Function_Specs.doc)
- test script* (Filename: < Surname_Firstname>_Test_Script.doc)
- sample question files
- sample bonus question files

Checklist (Place all of this in a short brown envelope*)

- ☐ CD (placed in case and labeled with name and section)
  - ☐ source code (Filename: <Surname_Firstname>.c)
  - ☐ Executable
  - ☐ function specifications** (Filename: < Surname_Firstname>_Function_Specs.doc)
  - ☐ test script** (Filename: < Surname_Firstname>_Test_Script.doc)
  - ☐ sample question files
  - ☐ sample bonus question files

- ☐ Print-out of the function specifications
- ☐ Print-out of the test script

* The brown envelope should be labeled with the following:

Name:	
Section:	Sxxx
Submitted To:	Mr. Richard Thomas Cruz
Received by:	
Date/Time Rcvd.:	

** - See separate file for deliverable format.

Send also to **richard.cruz@dlsu.edu.ph** account a copy of your source code & function specification by the deadline.

Filename: <Surname_Firstname>.c & < Surname_Firstname >.doc
E-mail Subject: COMPRO2 MP <Surname Firstname Block>

Being unable to show up on time during the demo schedules, or being unable to answer convincingly the questions during the demos, will merit a grade of 0.0 for the course. This is an individual project. Any form of cheating (copying any part of other's work, etc.) will merit a grade of 0.0 for the course & a discipline case.

VII. References

Wheel of Fortune gameplay and rules - http://www.ehow.com/way_5390187_wheel-fortune-rules.html