ECGR 2103

Course Project Guidelines

Spring 2019

DR. SMITH-ORR & Dr. RHOADES
2-27-2019

Instructions

Each team is to select one of the following games to develop for your project. The required contents of each program are listed below.

Team Contract

Within one week of receiving your teams your group must meet to develop your team contract. A copy of the contract should be submitted **by EACH member of the team** on Canvas and will contribute to your grade. The team contract should include (see samples attached):

- Plan for scheduling meetings or determined meeting dates
- Each members responsibilities and any rules established by the group (be as specific as possible)
- Project selected by the group
- Team Name

Peer Evaluations & Challenges Summary

1. There will be two surveys that will need to be completed by each member of your team. The survey will ask you about your performance as well as the performance of your teammates throughout the project. Failure to complete the survey will result in a loss of points. These survey responses may also impact your grade if it is found that you have not adequately contributed to the project.

Logic Plans

Logic Plans (Flow Chart): All topics necessary for this topic have not yet been covered in the course. However, as a way of establishing good logic/engineering skills you must establish a plan for your project at several points throughout the course of the project. The plan should include a flowchart and can also include a written description of your plan. You can use this resource to learn about flowcharting. Flowchart can be done through this website or sketched by hand.

a. March 15th - initial plans
b. April 3rd - revised plans
c. Final overview of the project to be presented with the final project
    i. Presentation will demonstrate the evolution of the project plan

Final Evaluation

Each team will give a 10-minute presentation explaining the major aspect of their program and decisions that were made in its development. Each member must have a speaking part in the presentation. A demo of the program should also be given during the presentation. Each member should understand all portions of the program.

A portion of the feedback for the presentation will be evaluated by your peers. Each group will be asked a minimum of 3 questions from the audience.

Grading

	5-4 Points	3-2 points	1-0 points	Score
1. User Interface	Interface is neat and well organized	Interface is reasonably neat	Interface is untidy and not organized	
2. Run without error	Runs smoothly and without glitches	Runs with some glitches	Still contains error and warning	
3. Appropriate functionality	Program meets the original question & gives correct output	Program follows the original partially or gives some incorrect output	Program is completely incorrect	
4. Includes Necessary Requirements	Includes 4-5 of the required components of the assigned project	Includes 2-3 of the required components of the assigned project	Includes 0-1 of the required components of the assigned project	
5. Clear Presentation	Presents about the whole project in an organized manner	Presents about the program only and in an unclear manner	Presentation is unclear and not understandable	
6. Questions	Presenters are able to answer each question and demonstrate their understanding of the program	Presenters answer some of the questions unclearly or incorrectly and show some understanding	Presenters are unable to clearly answer the questions and show little understanding	
Total Score				
Creativity (Bonus)	Goes beyond the requirements in several innovative ways that uniquely enhance the game and makes it more entertaining for the user	Goes beyond the requirements in 3-4 unique features	Goes beyond the requirements in 1-2 simple features	

Deadlines

Team Contract – 3-1-19 by 11:59 pm on Canvas

Logic Plans – 3-15-19; 4-3-19; Final in the presentation

Peer Reviews – 3-20-19; 4-16-19 (Emailed Link from Peer Evaluator)

Final Project must be submitted by midnight to Canvas on 4-14-19

- Presentations will take place between Apr. 15 & Apr. 29

-

Project Options

Whac-A-Mole

This program:

1. Develops a board of 'holes', one of which contains a 'mole' that must be selected by the user in order for the mole to be hit.
2. There will be three levels to the game that will automatically be entered based on player performance.
   a. Level 1 must be a 2x2 board
   b. Level 2 must be a 3x3 board
   c. Level 3 must be a 4x4 board
3. The options for the holes will be specific keys on the keyboard (i.e. keys Q,W,A,S for Level 1, Q,W,E,A,S,D,Z,X,C for Level 2, etc)
4. The location of the mole should change based on a specific time set in the program.
5. Progression to the next level should be based on if the player can progress successfully through each round.
6. Output the players score.

Program should include:

- Functions
- Array
- Loop
- Branching
- User Input
- Random Number Generator

Program Format:

- Directions & Team Name
- Comments

Each Participant should be able to:

- Explain each line of code

Assumptions:

The player can only fill in from the bottom of the board and on top of other plays.

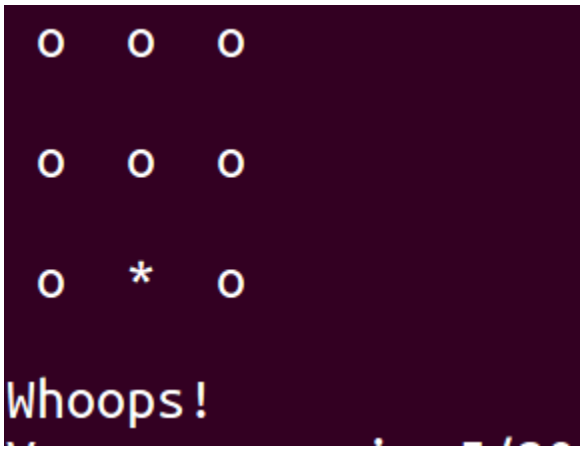
Creative Liberties:

- Symbols used for the board
- Keys used for play
- Time between appearance of each mole
- Requirements to progress through each level

- Layout & player interactions

Requirements for the project: **Whac-A-Mole**

1. Generate the 3*3 grid with '*' for the place where it occurs and 'o' for other places.
   For example



2. Accept the input from keys from your keyboard. The input is only one key.

	O O O		Q W E		
(on the screen)	O * O	→	A S D	(your keyboard)	
	O O O		Z X C		

3. Count the correct number of hits and calculate the hitting ratio. The results should be shown.

4. The hitting ratio is 100%, then show "You must be a genius!"
   The hitting ratio is between 80% and 100%, then show "You must be a master!"
   The hitting ratio is between 50 and 80%, then show "You could do better!"
   The hitting ratio is below 50%, then show "Not good..."

Chutes & Ladders (Build From Scratch)

In this game of Hangman:

Based on the board provided below, develop a program that mimics chutes and ladders. When a player rolls the dice they will move that number of spaces. If they land on a space at the bottom of a ladder they may climb the ladder to the corresponding space. If they land on the top of a chute, they must slide down to the corresponding space.

1. This is a two-player game.
2. The game will inform the user if they have rolled too high of a number to win the game and tell them that number. .
3. At any three point in the game there are tiles that will force the user to answer a question in order to roll their next turn. These must be C++ related questions.
4. In order to win the game the player must spin the exact number necessary to land at the end.

The program should include:

- Loop
- Function
- Array
- Rand
- Branching

Program Format:

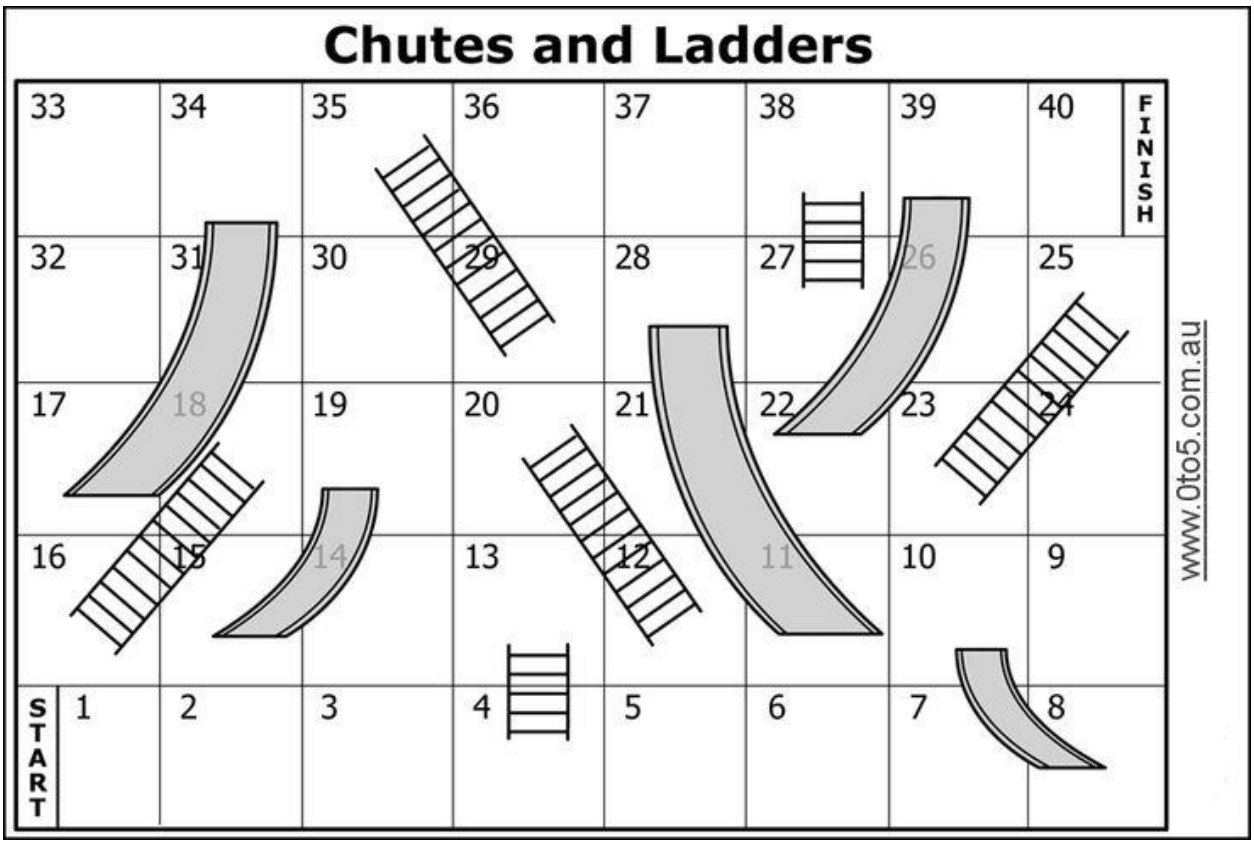
- Directions & Player Name
- Comments

Each Participant should be able to:

- Explain each line of code

Creative Liberties:

- Number of die
- Location of the hidden questions
- How the board is displayed visually and the indications for chutes and ladders
- You can determine if you would like to show the movement of each piece through the board or you can simply display the board at the beginning of each roll


Chutes and Ladders
33
34
35
36
37
38
39
40
FINISH
32
31
30
29
28
27
26
25
17
18
19
20
21
22
23
24
16
15
14
13
12
11
10
9
START
1
2
3
4
5
6
7
8
www.0to5.com.au

Battleship (Build from Scratch)

This program will include:

- Function
- Array
- Random number generator
- Loop
- Branching

Creative Liberties:

- How hits are displayed
- Display of the board
- Number of Boards stored
- Locations and sizes of ships
- Directions for the user

Battleship

The Navy needs your help! They are in desperate need for a suitable simulator that can be used to train new sailors. You and your team have been tasked with developing a new strategy based game that the sailors can use to hone their skills. You recall that you once played a game called Battleship as a kid.

For those that had the misfortune of not playing the game here is a synopsis:

1. This is typically a two player game each person has a grid that is similar to the images below where the columns are denoted by a letter and the rows a number

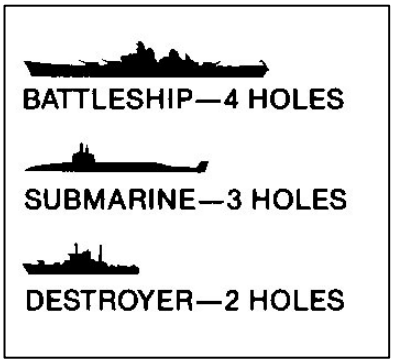
Player 1 Board

	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						

Player 2 Board

	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						

2. Each person has a fleet of three ships (Battleship, Submarine, and Destroyer). Each of the ships are assigned a number of holes. The image below illustrates the breakdown of the number of holes for each type of ship.



3. Each player will randomly distribute the three ships. The player is allowed the place ships horizontally or vertically but not diagonally. An example placement of both player's ships is shown below

Player 1 Board

	A	B	C	D	E	F
1	0	1	1	1	0	1
2	0	0	0	0	0	1
3	0	0	0	0	0	0
4	1	1	1	1	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0

Player 2 Board

	A	B	C	D	E	F
1	1	0	0	0	0	0
2	1	0	1	1	0	0
3	1	0	0	0	0	1
4	1	0	0	0	0	1
5	0	0	0	0	0	1
6	0	0	0	0	0	0

4. Each player takes a turn by giving a row and column combination to try to sink the opposing persons ship(s). After each player call the opposing player indicates a correctly guessed row and column (R & C) combination that contains part(s) of a ship by saying the word "Hit" (User gets unlimited turns so long as they continue to guess correct R & C combination that result in a "Hit") or "Miss" if the player's R & C combination did not contain any part of a ship(s). An example of a "Miss" would be if player one gives the R & C combination of "B1", this would result in player 2 saying "Miss" since that combination did not impact any of their fleet. Likewise, a "Hit" example would be if the same player would have given the R & C combination of "A1". Player 2 would have said "Hit" and player 1 would have been given another turn to try to give a R & C combination that would result in another "Hit".

5. The game continuous until one player has sunk all the other players' fleet (three ships). At the conclusion of the game the looser concedes defeat and the two players shake hands.

Program Requirements

Your team's task is to replicate this game given the following requirements:

1. Must use at least 2 functions
2. Must use two 2D arrays to act as each player's board.
3. The user must be able to choose their difficulty level (similar to minesweeper):

"e" for easy mode (5 x 5 program board size)

"m" for medium mode (6 x 6 program board size) *as shown*

"h" for hard mode (7 x 7 program board size)

4. Must use at least 3 pre-populated player boards per level (aka there should be 3 player 1 and player 2 board combinations that you manually develop)
5. Must use random number generator (to randomly choose populated game board for each player)
6. Must refresh terminal screen after each player's move
7. Must display "*" on every location on both player's board at the start of each game.
8. Must display a symbol (i.e. "X") on every "Hit" made by the player on each player's board.
9. Must accept user input for Row & Column combination(s)
10. Must comment code and compile program in Linux

Additional information (aka Hints):

- Use a function to populate each board levels choice

- Store each player's board as 1s and 0s. For the example player boards shown above they would translate to:

Player 1 Board

	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						

Player 2 Board

	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						

- Take the user input (B3) as an example and translate that into an index for the 2D arrays (i.e. B3 translates to (2,1) in the 2D array.

- "and" the users entered location with your 1s and 0s 2D array at the same location (The result will be a 1 if it is a hit or 0 if it was a miss.

Below is an example input and output for this project (based on the example boards shows above). →

Full program example output (and input)

>> Welcome to the Battleship Game Simulator! Please choose your level of difficuly (e = easy, m = medium, h = hard)

>> m

>> You have chosen a medium level game

>> Please enter in Player # 1's Name:

>> John

>> Please enter in Player # 2's Name:

>> Fred

>> Here is the current Gameboard:

John's Board

	A	B	C	D	E	F
1	*	*	*	*	*	*
2	*	*	*	*	*	*
3	*	*	*	*	*	*
4	*	*	*	*	*	*
5	*	*	*	*	*	*
6	*	*	*	*	*	*

Fred's Board

	A	B	C	D	E	F
1	*	*	*	*	*	*
2	*	*	*	*	*	*
3	*	*	*	*	*	*
4	*	*	*	*	*	*
5	*	*	*	*	*	*
6	*	*	*	*	*	*

>> John's Turn, please enter a Location:

>> B2

>> I am sorry John that was a MISS!

>> Here is the current Gameboard:

John's Board

	A	B	C	D	E	F
1	*	*	*	*	*	*
2	*	*	*	*	*	*
3	*	*	*	*	*	*
4	*	*	*	*	*	*
5	*	*	*	*	*	*
6	*	*	*	*	*	*

Fred's Board

	A	B	C	D	E	F
1	*	*	*	*	*	*
2	*	*	*	*	*	*
3	*	*	*	*	*	*
4	*	*	*	*	*	*
5	*	*	*	*	*	*
6	*	*	*	*	*	*

>> Fred's Turn, please enter a Location:

>> F1

>> Congratulations Fred that was a HIT!

>> Here is the current Gameboard:

John's Board

	A	B	C	D	E	F
1	*	*	*	*	*	X
2	*	*	*	*	*	*
3	*	*	*	*	*	*
4	*	*	*	*	*	*
5	*	*	*	*	*	*
6	*	*	*	*	*	*

Fred's Board

	A	B	C	D	E	F
1	*	*	*	*	*	*
2	*	*	*	*	*	*
3	*	*	*	*	*	*
4	*	*	*	*	*	*
5	*	*	*	*	*	*
6	*	*	*	*	*	*

>> Fred's Turn, please enter a Location:

>> F2

>> Congratulations Fred that was a HIT!

>> Fred sank John's Destroyer!

>> Here is the current Gameboard:

```
          John's Board                  Fred's Board
     A  B  C  D  E  F            A  B  C  D  E  F
1    *  *  *  *  *  X         1  *  *  *  *  *  *
2    *  *  *  *  *  X         2  *  *  *  *  *  *
3    *  *  *  *  *  *         3  *  *  *  *  *  *
4    *  *  *  *  *  *         4  *  *  *  *  *  *
5    *  *  *  *  *  *         5  *  *  *  *  *  *
6    *  *  *  *  *  *         6  *  *  *  *  *  *
```

>> Fred's Turn, please enter a Location:

>> F3

>> I am sorry Fred that was a MISS!

>> Here is the current Gameboard:

```
          John's Board                  Fred's Board
     A  B  C  D  E  F            A  B  C  D  E  F
1    *  *  *  *  *  X         1  *  *  *  *  *  *
2    *  *  *  *  *  X         2  *  *  *  *  *  *
3    *  *  *  *  *  *         3  *  *  *  *  *  *
4    *  *  *  *  *  *         4  *  *  *  *  *  *
5    *  *  *  *  *  *         5  *  *  *  *  *  *
6    *  *  *  *  *  *         6  *  *  *  *  *  *
```

>> John's Turn, please enter a Location:

Etc... until the game is finished...

References

Picture:

[1] https://cf.geekdo-images.com/images/pic288374.jpg

[2] http://petevsdanbattleship.com/game-rules/

Simon Game Assignment:

Welcome to the 80's! If you want to party hardy and then you need to check out this totally tubular game! This memory game will see if you have what it takes to take it to the max!

If you have never heard of this gnarly game then kick back and watch this rad video on the tube... YouTube that is...

https://www.youtube.com/watch?v=debcf-S591U

If you are still unclear about the basics of the game it essentially is a visual memorization game that repeats a pattern for the user to mimic (based and names from the kids game of Simon says...). The game always starts with one color and then continues to add one additional color to the sequence. The game continues until the user cannot recall and repeat back the entire sequence in order.

Take note that there are four evenly divided colors on the games surface (red [R], Green [G], Blue [B], and Yellow [Y]).

This can easily be translated into the terminal (see example game board below)

```
-----------------------------------
-                 -               -
-      GGGG       -     RRRR      -
-     G    G      -     R   R     -
-    G             -     R   R     -
-    G   GGGG      -     RRRR      -
-    G      GG     -     RR        -
-     G     G      -     R R       -
-      GGGGG       -     R  R      -
-                 -               -
-----------------------------------
-                 -               -
-    Y       Y    -     BBBB      -
-     Y     Y     -     B   B     -
-      Y   Y      -     B   B     -
-       YYY       -     BBBB      -
-        Y        -     B   B     -
-        Y        -     B   B     -
-        Y        -     BBBB      -
-                 -               -
-----------------------------------
```

Since the idea behind the game is to repeat a pattern you will need to mimic this behavior in the terminal. To aid in the process you will need to use an additional library and some functions associated with that library to clear the screen (see below)

```
#include <unistd.h>
```

```
usleep(1000000);
system("tput clear");
```

Your team will need to investigate how these functions operate.

Your teams' task is to replicate this game given the following **requirements**:

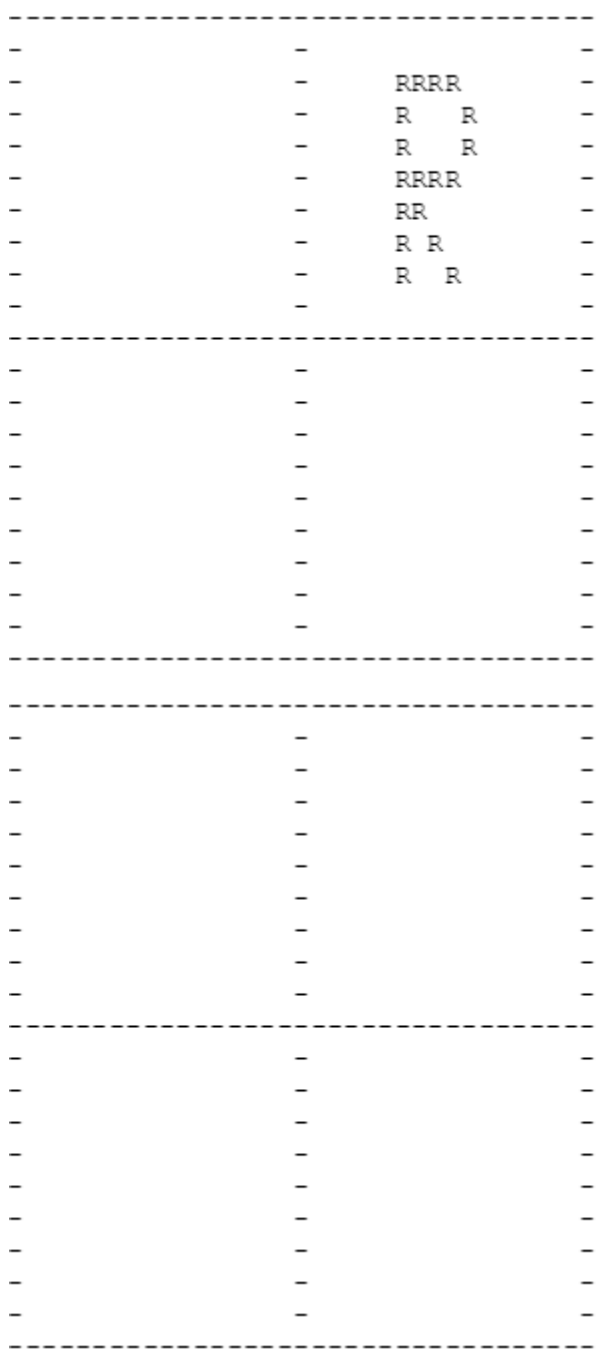
1. Must use at least 2 user defined functions
2. Must use two arrays to act as each player's board.
3. The user must be able to choose their difficulty level (rate of screen refresh):
   "e" for easy mode (shows each letter sequence for 1 second)
   "m" for medium mode (shows each letter sequence for 0.5 seconds)
   "h" for hard mode (shows each letter sequence for 0.25 seconds)
4. Must use random number generator (to randomly make up the Simon sequence)
5. Must refresh / clear terminal screen to mimic the actual game's flashing lights.
6. Must keep up with the users score and display their final score, along with how many rounds the user was able to complete before exiting the game.
7. Must comment code and compile program in Linux

Below is an example round of game play (recall that the screen will transition from the empty board to a board with one of the letters in sequence:

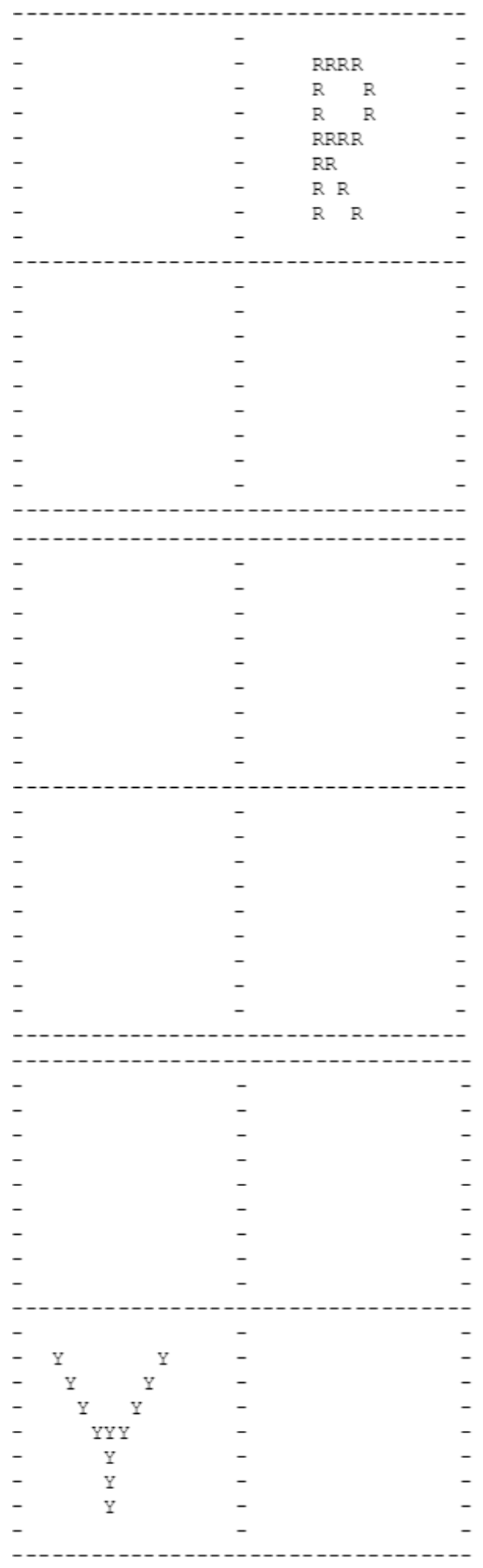
>> Please enter your level of difficulty:

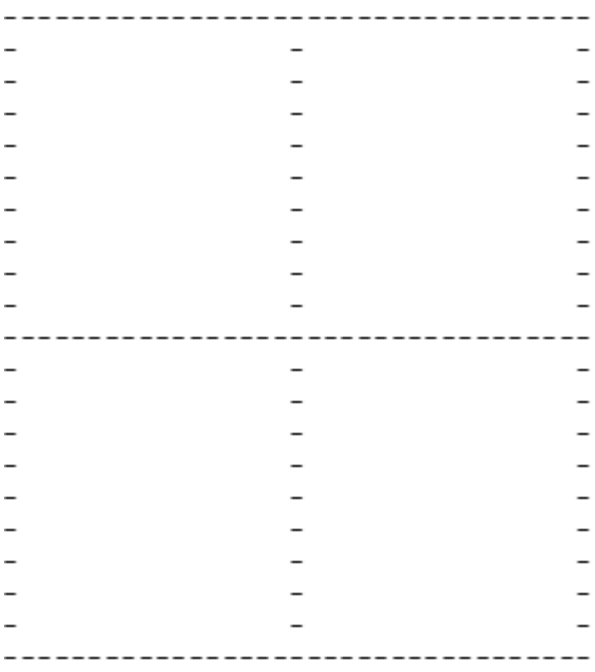
'e' for easy
'm' for medium
'h' for hard

level:e



>> Please enter in the character sequence you observed on the screen
R


RRRR
R R
R R
RRRR
RR
R R
R R
Y Y
Y Y
Y Y
YYY
Y
Y
Y



```
>> Please enter in the character sequence you observed on the screen
   RR

>> Sorry the combination was not correct your final score is 1
>> You made it to round number 2
>> Game Over!
```

References

Picture:

[1] https://boardgamegeek.com/image/288369/simon