CSC 340.02 - Spring 2022 - Assignment 2
Due: Friday 03/11/2022 11:55pm
Grade: 100pt - 20%
Late submission: 2 days allowed. Grades will be deducted 20% each day.

<u>Guidelines for the assignment submissions:</u>

What to submit?

- One zip file:
    - Name: `<First Name><Last Name>-Assignment-##.zip`
    - Source code should include only files that we create and edit (*.cpp and/or *.h)
    - One file for part 1. One file for part 2. One directory for part 3

How to submit?
iLearn, Assignment Submission section

1. Pascal triangle (15pt)

Write a program to take an integer n (with n<10) as the command line argument input and output the Pascal triangle for that n number of lines. For this exercise, you cannot use any standard library containers.

Input: 6 (command line argument)
Output:

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
```

2. Recursion (15pt)

Implement a recursive function to determine whether a string is palindrome or not, considering only alphanumeric characters and ignoring cases. Write a driver program to test your function by asking the user to input a string and determine if it’s palindrome.

Example:

```
Input: s = "A man, a plan, a canal: Panama"
Output: true
Explanation: "amanaplanacanalpanama" is a palindrome.
```

```
Input: s = "race a car"
Output: false
Explanation: "raceacar" is not a palindrome.
```

3. Adaptive memory game

Your program outputs random words (from a predefined set of words) on screen, waits for 1 seconds, then clears the screen, takes the user's input and validates the input.

Your program starts with 2 words originally and increases the number of words depending on how well the users play the game.

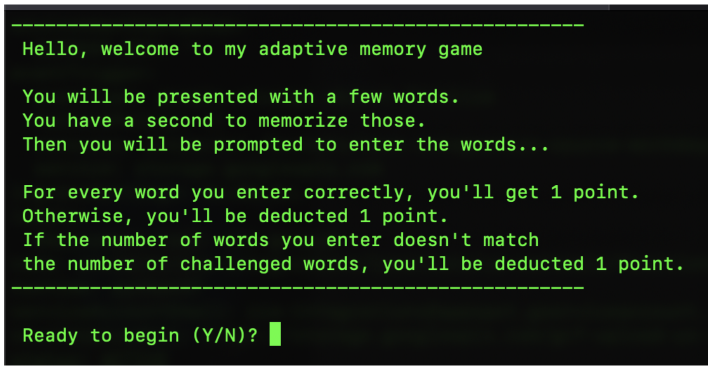
Your program keeps track of the user's score. For every word they respond correctly, they get 1 point, otherwise they get deducted 1 point. If the number of words they enter do not match the number of challenged words, they get deducted 1 point.

For every 4 points the user gets, they will be bumped to the next level. For every next level, they will have one extra random word in the output.

Your program exits when the user types "q" or when they have more than 10 points.

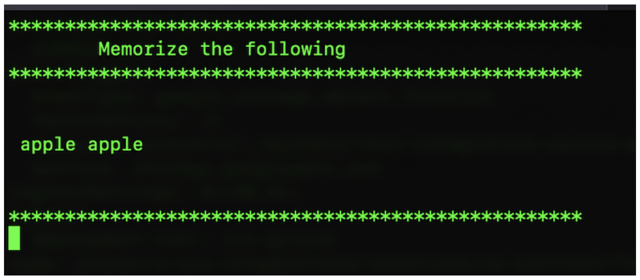
You can use standard library containers for this exercise. If you'd like, you can use the included set of words and snippets for implementing sleep function and clear function.

- Sample output of the game



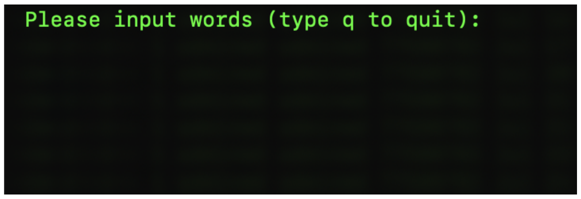
```
----------------------------------------------------
 Hello, welcome to my adaptive memory game

 You will be presented with a few words.
 You have a second to memorize those.
 Then you will be prompted to enter the words...

 For every word you enter correctly, you'll get 1 point.
 Otherwise, you'll be deducted 1 point.
 If the number of words you enter doesn't match
 the number of challenged words, you'll be deducted 1 point.
----------------------------------------------------

 Ready to begin (Y/N)?
```

Step 1: when the game starts, type Y to continue, otherwise, quit.



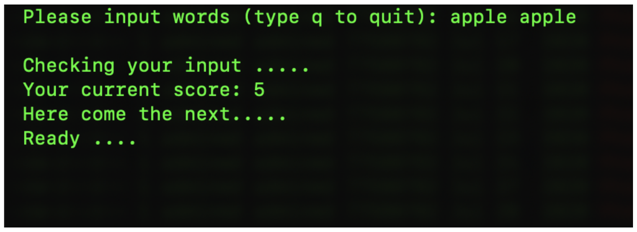
```
****************************************************
        Memorize the following
****************************************************


 apple apple


****************************************************
```

Step 2: display words, and sleep 1 second



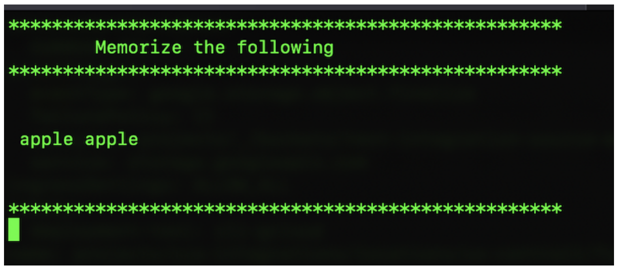
```
Please input words (type q to quit):
```

Step 3: clear screen, and let player enter words



```
Please input words (type q to quit): apple apple

Checking your input .....
Your current score: 5
Here come the next.....
Ready ....
```

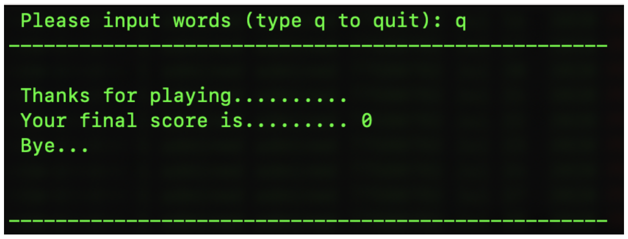
Step 3: After the player enters the words, your program should check the result and calculate the score. For each of the above lines, sleep 1 second.



```
****************************************************
        Memorize the following
****************************************************


 apple apple


****************************************************
```

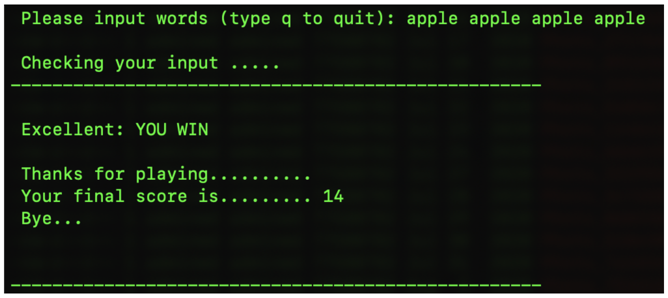
Step 4: clear screen and display the next set of random words.

And repeat step 3 and step 4 until the player quits or wins the game.



```
 Please input words (type q to quit): q
---------------------------------------------------

 Thanks for playing.........
 Your final score is......... 0
 Bye...


---------------------------------------------------
```

Step n: The player types `q` to quit if they decide to.



```
 Please input words (type q to quit): apple apple apple apple

 Checking your input .....
-------------------------------------------------

 Excellent: YOU WIN

 Thanks for playing.........
 Your final score is......... 14
 Bye...


-------------------------------------------------
```

Final step: The player wins if they have more than 10 points. Game exits automatically.

- Grading criterions: (70pt)
  - Design document (12pt): In a pdf file, describe your strategy of handling this problem. Why do you pick this strategy? If you choose to implement this using object oriented programming, describe your interface. If you choose to implement this using functional programming, describe your functions. Your design document can be organized into the following:
    - Motivation and problem statement
    - Requirements
    - Implementation plan: list of functionality and success metrics
    - Future improvements
  - Readme file (3pt): In a readme.txt file, write a short summary of the program, how to compile your code and how to run it.
  - Implementation: (50pt)
    - Readability + good comments (10pt) - follow Google Style Guide to format your code https://google.github.io/styleguide/cppguide.html#Formatting
    - Meet specifications: (40pt)
      - Correct workflow
      - Output challenged words
      - Input words
      - Adaptive difficulty level
      - Exit from player's input or player wins the game
  - Enhance the game (5pt): How would you enhance this game? Describe a few ideas and your solutions to implement those in your design doc.