San José State University
Department of Computer Science

CS 144
Advanced C++ Programming

Spring 2019
Instructor: Ron Mak

Assignment #9

Assigned: Tuesday, April 9
Due: Tuesday, April 16 at 8:30 AM
Canvas: Assignment 9. GUI-Based Rock-Paper-Scissors Game
Points: 100

GUI-Based Rock-Paper-Scissors Game

Use **wxWidgets** to create a GUI-based version of a Rock-Paper-Scissors game program. Required features:

- For each round, there is a way for the human player to choose rock, paper, or scissors. The computer makes a random choice. Display the human's and the computer's choices.
- Display who won each round (or was it a tie).
- Display a running count of human wins, computer wins, and ties.
- The default is 20 rounds per game. Provide a way to change that number.
- Required menu commands:
    - About
    - Exit
    - Start a new game

Written report

In a short report, describe:

- What events does your game application generate?
- How did you use callback functions to handle the events?

Include a screen shot of your GUI in your report.

What to turn in

Make a zip file of all your C++ source files and your report.

Submit it into Canvas: **Assignment #9. GUI-Based Rock-Paper-Scissors Game**

Rubric

Your program will be graded according to these criteria:

Criteria	Max points

• **GUI components:**
  ○ Which round
  ○ A way for the user to enter a choice for each round.
  ○ The computer's choice for the round.
  ○ Who the winner is (or is it a tie) of the round.
  ○ The number of human and computer wins, and the number of ties.
• **Menu items:**
  ○ About
  ○ Exit
  ○ Start a new game
• An interactive way to change the number of rounds per game.
• The grader is able to play several games.
	• **60**
	  ○ 10
	  ○ 20
	  ○ 10
	  ○ 10
	  ○ 10
	• **15**
	  ○ 5
	  ○ 5
	  ○ 5
	• **10**
	• **15**

Extra credit: RPS Game with simple machine learning

Human players of the Rock Paper Scissors game try to develop **strategies** to beat their opponents. Therefore, humans generally do not make random choices. Instead, their choices exhibit **patterns** that a computer can discover and exploit using a simple **machine learning** algorithm.

The computer's ML choice algorithm

Continuously record the last *N* choices between the human and the computer player. Throw out the oldest choice in order to add a new one. For example, suppose *N* = 5 and during the game, the recorded choices are (the human's choices are underlined):

PSRSP

The last choice was made by the human, and it was paper.

For each recorded sequence that ends with the human's choice, the computer should store how many times that sequence has occurred (each sequence's frequency).

For example, for *N* = 5, some of the stored sequences and their frequencies may be (in no particular order, the human choices are underlined):

RSPSR:1, **SPRPP**:3, **RSPRS**:2, **RSPSS**:4, **RSPSP**:3, **SSRPP**:2

Suppose that in the previous round, the human chose rock and the computer chose scissors. In the current round, the human chose paper and the computer chose scissors. Therefore, the last four choices were **RSPS**.

The computer can predict that the human will most likely next choose scissors, since **RSPSS** appears more times (4) than **RSPSR** (1, predict rock) and **RSPSP** (3, predict paper) in the stored frequencies. Therefore, the computer should choose rock to beat

the human's predicted choice of scissors. After the human makes a choice, update the appropriate frequency.

If a sequence is not in the store, then the computer can make a random choice.

As the computer plays more games and stores more frequency data, it becomes increasingly more capable of predicting the human's next choice. Over the long haul, the computer will become very difficult for humans to beat.

Example screenshot
This screenshot includes the extra credit. The computer made the correct prediction that the human would choose paper for this round and therefore it chose scissors to win.