CS 101
Fall 2013
Program Assignment 3
Algorithm due **Sunday, Sept. 22.**
Program due **Sunday, Sept. 29.**

Rock-Paper-Scissors is an ancient and well-known game, sometimes used as a sort-of-random selection method. Of course, the game isn't truly random, since an intelligent player can observe and exploit non-random behavior of their opponent. For example, if you notice that your opponent most often chooses Paper, then you can choose Scissors (which beats Paper) in an attempt to win.

Rules of the game:
Each player chooses a weapon: Rock, Paper, or Scissors, and reveals it to the other at the same time. The winner is determined by simple rules:

- Rock breaks Scissors (Rock wins)
- Scissors cuts Paper (Scissors wins)
- Paper covers Rock (Paper wins)
- If both players choose the same thing, it's a tie, and neither wins.

This program will require you to:

- Get input from the keyboard, and perhaps do some simple string manipulation
- Print results, perhaps by building up a string from different building blocks based on program events.
- Use branching (if/elif/else) and loop control (while) logic
- Use Boolean logic
- Track the state (history) of the game and use it to make decisions.

Your program will let a user play Rock – Paper – Scissors with the computer. It will work as follows:

- Your program will select its 'weapon' (Rock, Paper, or Scissors), but NOT display it yet.
- Ask the user for their choice and tet input from the user: 'r' or 'R' for Rock, 'p' or 'P' for Paper, 's' or 'S' for Scissors, 'q' or 'Q' for Quit. Anything else should print a brief error message.
- Compare the results and announce the winner of the round: the player wins, the program wins, or it's a tie.
- Continue playing until the user chooses to quit.
- When the user chooses to quit, provide a summary of the game: How many rounds were played, how many rounds the player won, lost, and tied, and how many times each weapon was chosen.

Your program should select the weapon most likely to beat the user, based on the user's previous choice of weapons. For instance, if the user has selected Paper 3 times but Rock and Scissors only 1 time each, the computer should choose Scissors as the weapon most likely to beat Paper, which is the user's most frequent choice so far.

To accomplish this, your program must keep track of how often the user chooses each weapon. Note that you do not need to remember the order in which the weapons were used. Instead, you simply need to keep a count of how many times the user has selected each weapon (Rock, Paper or Scissors). Your program should then use this playing history (the count of how often each weapon has been selected by the user) to determine if the user currently has a preferred weapon; if so, the computer should select the weapon most likely to beat the user's preferred weapon. During rounds when the user does not have a

single preferred weapon, the computer may select any weapon. For instance, if the user has selected Rock and Paper 3 times each and Scissors only 1 time, or if the user has selected each of the weapons an equal number of times, then there is no single weapon that has been used most frequently by the user; in this case the computer may select any of the weapons.

Development hints:

- There's a string method called .lower(), that returns a lower-case version of the original string. 'ABC'.lower() = 'abc'. This can make managing the input a bit simpler.
- Do all the standard startup things. Create a new file. Put your comments in at the top, save it.
- Now you need to break the problem down into parts. Read the description and identify the subtasks that need to be solved. For example, one subtask would be to get proper user input. Mark in the empty program, using comments, all the subtasks you need to solve.
- Now address one subtask; let's say you choose getting user input to start. Do this in stages as well. Can you:
  - Prompt for and get a choice (a string) from the user?
  - Once you can do that, can you repeatedly prompt for a character until you see a ‘q’ or ‘Q’ for quit?
  - Once you can do that, can you check for “legal” character responses from the user, and print an error message when an illegal response is given?
  - Next, can you check for legal responses that are in both upper and lower case?
- Once you can do all that, move on to the next subtask.
- Remember, save the file and run it frequently! As soon as you make any significant change, test it. It will make debugging the program easier.

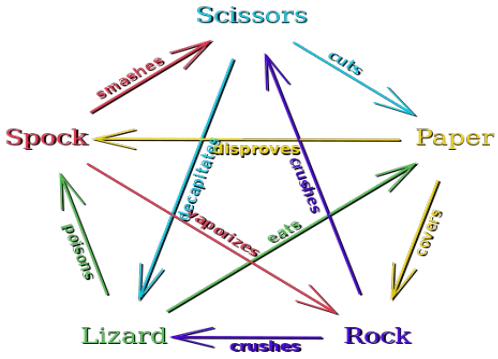
Extra credit 1: For 5 points extra credit, add a new option: 'h' or 'H' for Help. If the user chooses Help, print a brief summary of all valid inputs, and which weapons defeat the others. If you do this, then at the beginning of the program, print a brief message telling the user to enter 'H' or 'h' for help. Obviously, choosing Help does not affect any of the game statistics.

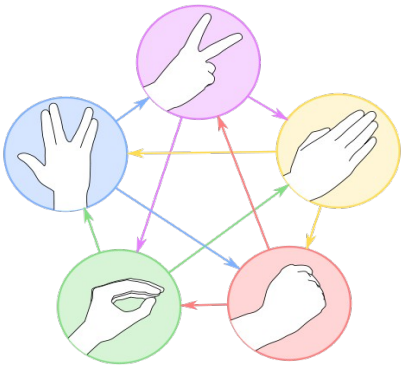
Extra credit 2: For another 5 points extra credit, implement the 5 weapon version (Rock, Paper, Scissors, Lizard, Spock) popularized on *Big Bang Theory.* This version introduces 2 new weapons, Lizard, and Spock:

- Rock (R) breaks Scissors, crushes Lizard (Rock wins)
- Paper (P) covers Rock, disproves Spock (Paper wins)
- Scissors (SC) cuts Paper, decapitates Lizard (Scissors wins)
- Lizard (L) eats Paper, poisons Spock (Lizard wins)
- Spock (SP) breaks Scissors, vaporizes Rock (Spock wins)
- See diagram below for a summary.

If you do this, you should begin the game by telling the user that this is the 5-weapon version, providing a summary like the above, and modifying your expectations for input accordingly. This can be done in addition to Extra Credit 1, above.

Here's a diagram (source: Wikimedia Commons, from Wikipedia article on Rock-Paper-Scissors):





Yet Still More Extra Credit:

For 10 points extra credit, give the user the option of playing the 3-weapon or 5-weapon version, and play whichever game the user selects. This can be done in conjunction with Extra Credit 1 above, but not Extra Credit 2. (In other words, you can get 5 points extra credit for the help screen, or not; and either 5 points extra credit for a program that unconditionally plays the 5-weapon version, or 10 points extra credit for a program that gives the user the choice.)

Sample Run (basic program, no extra-credit):

```
>>> ================================ RESTART ================================
>>>
Choose your weapon! ==> r
You chose Rock
I chose Scissors
You win.
Choose your weapon! ==> r
You chose Rock
I chose Paper
I win.
Choose your weapon! ==> p
You chose Paper
I chose Paper
Neither of us win.
Choose your weapon! ==> p
You chose Paper
I chose Paper
Neither of us win.
Choose your weapon! ==> p
You chose Paper
I chose Rock
You win.
Choose your weapon! ==> p
You chose Paper
I chose Scissors
I win.
Choose your weapon! ==> p
You chose Paper
I chose Scissors
I win.
Choose your weapon! ==> r
You chose Rock
I chose Scissors
You win.
Choose your weapon! ==> r
You chose Rock
```

```
I chose Scissors
You win.
Choose your weapon! ==> hello
Bad input, try again.
Choose your weapon! ==> q
That was a good game!
We played 9 rounds.
You won 4 rounds, I won 3 and we tied on 2
You used Rock 4 times, Scissors 0 times, and Paper 5 times.
```