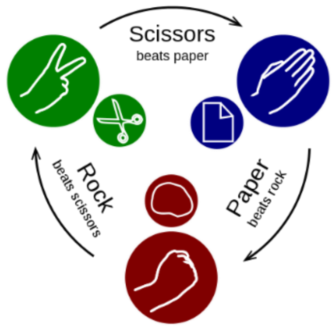
CS 150 Lab 10
Functions and Random Numbers

The objective of today's lab is to implement functions and random numbers in a simple game.

- Be sure your output looks exactly like the specified output.
- Be sure to submit the completed project to CS150-03 Lab by Friday at 5pm.
- Be sure to follow the coding standards and add comments to your code!

Lab 10

Write a C++ program in a project called **10_1_RockPaperScissors** that will simulate the game of rock paper scissors:

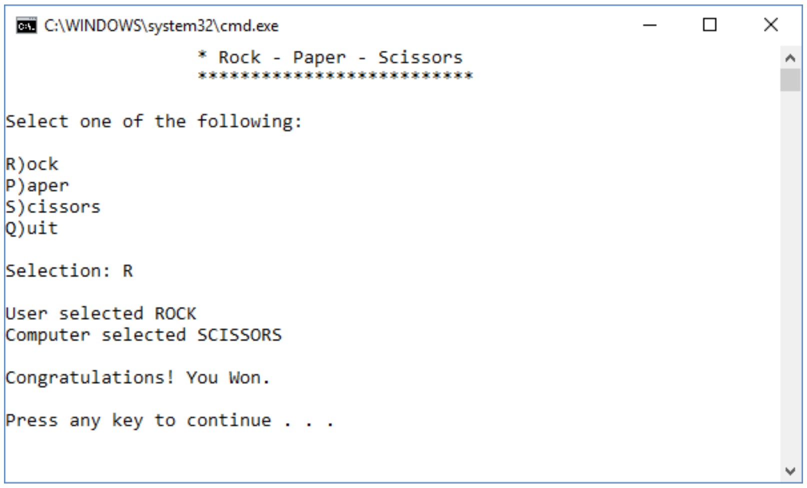


Source: Wikipedia

The players are the computer and the user. The user will select their choice from a menu, and the computer's choice will be randomly generated (it may be helpful for you to review how random numbers are generated from the lecture notes).

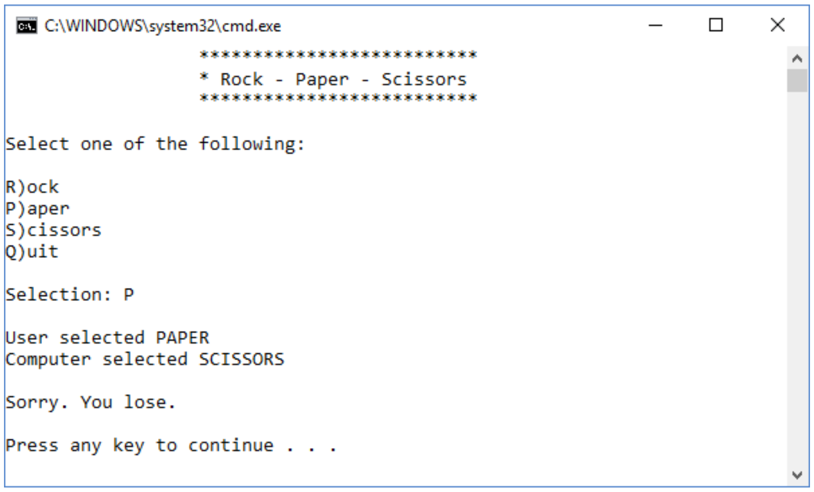
Here are several runs of the program:

Run 1:



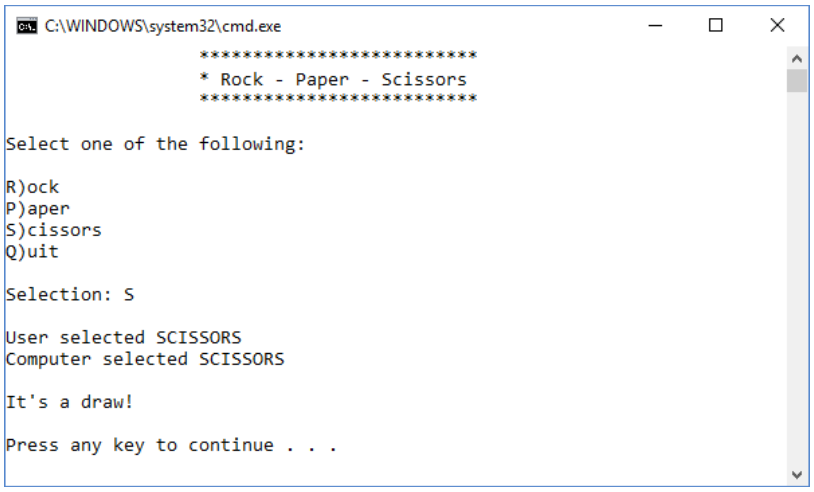
```
                  * Rock - Paper - Scissors
                  **************************

Select one of the following:

R)ock
P)aper
S)cissors
Q)uit

Selection: R

User selected ROCK
Computer selected SCISSORS

Congratulations! You Won.

Press any key to continue . . .
```

Run 2:



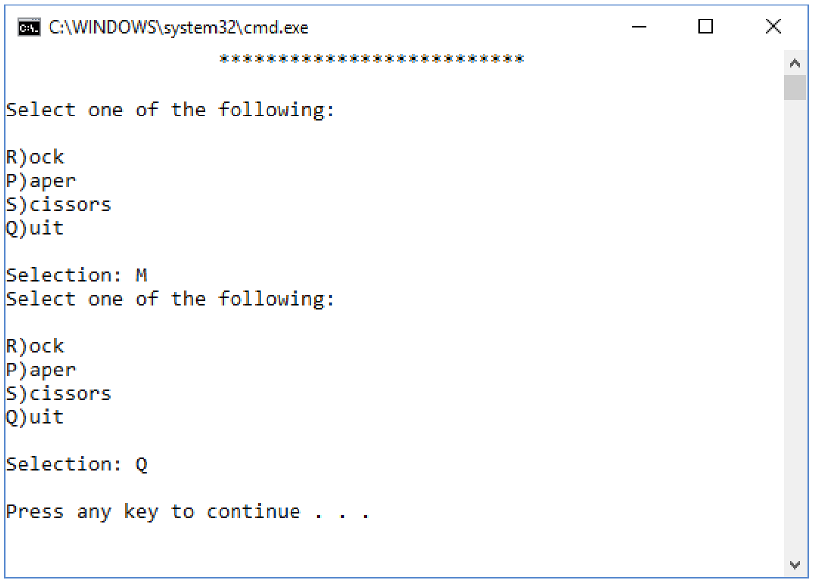
```
                  **************************
                  * Rock - Paper - Scissors
                  **************************

Select one of the following:

R)ock
P)aper
S)cissors
Q)uit

Selection: P

User selected PAPER
Computer selected SCISSORS

Sorry. You lose.

Press any key to continue . . .
```

Run 3:



```
                  **************************
                  * Rock - Paper - Scissors
                  **************************

Select one of the following:

R)ock
P)aper
S)cissors
Q)uit

Selection: S

User selected SCISSORS
Computer selected SCISSORS

It's a draw!

Press any key to continue . . .
```

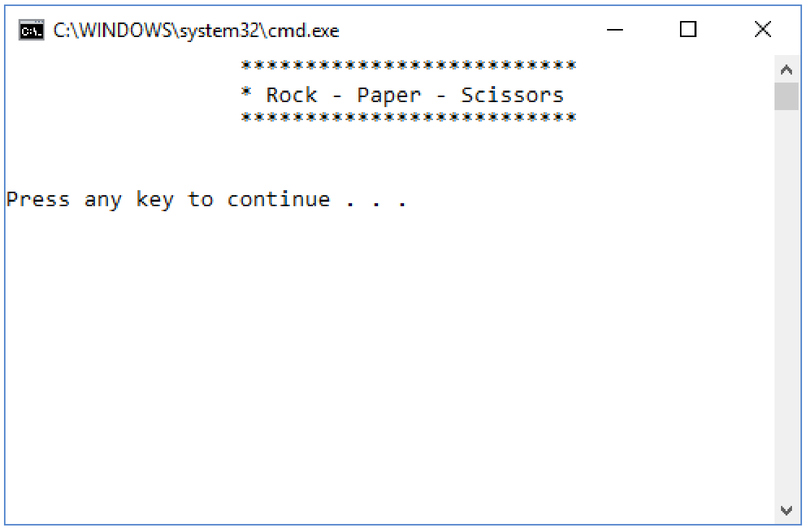
Run 4:



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

Select one of the following:

R)ock
P)aper
S)cissors
Q)uit

Selection: M
Select one of the following:

R)ock
P)aper
S)cissors
Q)uit

Selection: Q

Press any key to continue . . .
```

Step 0: In the public folder on Grace you will find a skeleton of the program named **10_1_RockPaperScissorsClean.cpp**. Create a new **main.cpp** file within your 10_1_RockPaperScissors project and copy this skeleton code into your project, update the program header comments, build, and execute it, and make sure that it compiles and runs without any errors or warnings.

As a verification, for the first run, your screen should look like this:



```
                  ***************************
                  * Rock - Paper - Scissors
                  ***************************


Press any key to continue . . .
```

This project requires seven functions, the first two of which have been completed for you, and the last five of which you will need to complete yourself. A complete description of each of the functions is in the comment header of the respective function in the skeleton program, but the prototypes are given below for reference:

Completed:

```
void printTitle (string title);
void displayMenu ();
```

You need to complete:

```
char getComputerSelection ();
char getUserSelection ();
void displaySelection (string player, char selection);
int determineWinner (char playerOneSelection, char playerTwoSelection);
void displayWinnerMessage (char userSelection, char computerSelection);
```

The way we would like to develop this project is to program each of the five functions above, one at a time, testing as we develop to make sure that each function is working for all valid input. Once you are convinced a particular function is working, you can move onto the next function.

Step 1: Write `char getComputerSelection ()`

First, generate a random integer between 1 and 3 inclusive. Using the global variables ROCK, PAPER, and SCISSORS, use this random integer and a switch statement to assign a value to the computer's selection, and for testing purposes please make the following associations:

1 corresponds to ROCK, 2 to PAPER, and 3 to SCISSORS.

TESTING: We want to seed the random number generator with seeds whose behavior we know from a working version of this program. To do this, comment out the following line in main:

```
srand (static_cast<unsigned> (time(static_cast<time_t *>(NULL))));
```

and below this line type: `srand (seed);`

When the value of `seed` in the expression above is set to a particular value below, it results in the corresponding computerSelection:

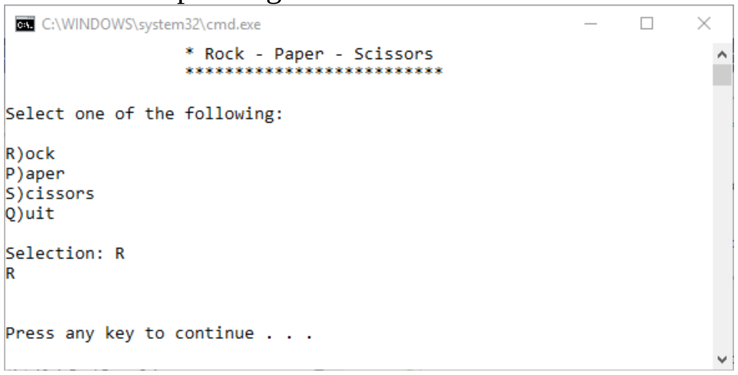
seed	**computerSelection**
100	S
150	R
200	P

To check these values, in main print to the screen the `computerSelection` that is returned from the function call `getComputerSelection ();`

When you are sure this function is working correctly**, uncomment the original srand statement,** delete the `srand(seed)` statement, and delete the command that prints the computer selection to the screen.

Step 2: Write `char getUserSelection ()` (NOTE: Be sure to VALIDATE!)

TESTING: Use a similar method to test this function as was used to test the function in Step 1, making sure that an 'R', 'P', 'S', or 'Q' is printed to the screen when you make the corresponding selection from the menu. **As an example:**



```
                    * Rock - Paper - Scissors
                    ***************************

Select one of the following:

R)ock
P)aper
S)cissors
Q)uit

Selection: R
R


Press any key to continue . . .
```

<u>**Show the instructor or TA your solution (Step 2)**</u>

When you are sure this function is working correctly, delete the command that prints the user selection to the screen.

Step 3: Write `void displaySelection (string player, char selection)`

Based on the parameter `selection`, use the global char constants ROCK, PAPER, and SCISSORS when determining which expression (ROCK_STRING, PAPER_STRING, or SCISSORS_STRING) to display. Note that you should only have **ONE** `cout` expression in this function.

TESTING: To test this function, you should comment out both the following lines in main:

```
computerSelection = getComputerSelection ();
userSelection = getUserSelection ();
```

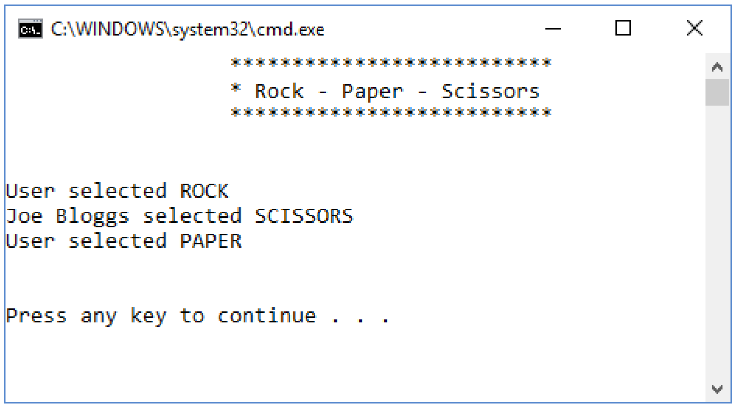
in addition to the lines that use these particular variables:

```
displaySelection ("User", userSelection);
displaySelection ("Computer", computerSelection);
displayWinnerMessage (userSelection, computerSelection);
```

Now, make very specific function calls to your `displaySelection` function in which you test a variety of `player` strings and valid `selection` characters. For example, when the following lines are placed in main

```
displaySelection ("User", ROCK);
displaySelection ("Joe Bloggs", SCISSORS);
displaySelection ("User", PAPER);
```

the output should be:



```
                   ***************************
                   * Rock - Paper - Scissors
                   ***************************


User selected ROCK
Joe Bloggs selected SCISSORS
User selected PAPER


Press any key to continue . . .
```

<u>Show the instructor or TA your solution (STEP 3)</u>

When you are convinced this function is working, uncomment the five lines above, and delete any test cases like those above that you may have created. Compile and run the program and make sure it is still functioning.

Step 4: Write `int determineWinner (char playerOneSelection, char playerTwoSelection)`

This is the heart of this program where you will implement the logic that decides whether you have won the game (use the global constants `WIN`, `LOSE`, and `DRAW`).

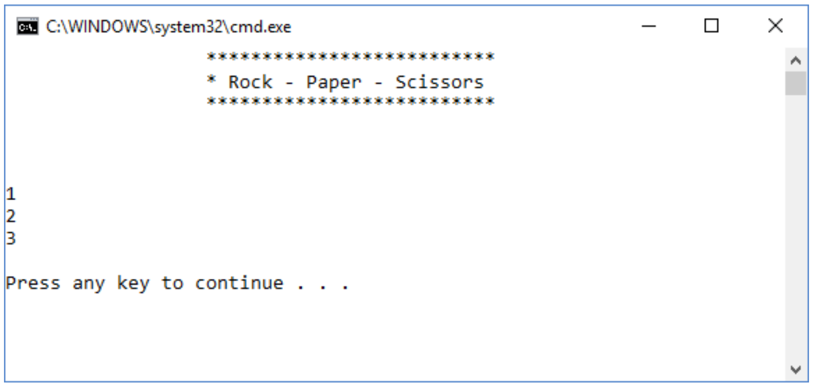
TESTING: We can test this function by commenting out the same five lines as we did in testing the function in the previous step:

```
userSelection = getUserSelection();
computerSelection = getComputerSelection();
displaySelection("User", userSelection);
displaySelection("Computer", computerSelection);
displayWinnerMessage(userSelection, computerSelection);
```

In the same way, make very specific function calls to your `determineWinner` function, printing the result to the screen for verification. For example, when the following lines are placed in main:

```
cout << determineWinner (ROCK, SCISSORS) << endl;
cout << determineWinner (PAPER, SCISSORS) << endl;
cout << determineWinner (ROCK, ROCK) << endl;
```

the output to the screen should be:



```
                  **************************
                  * Rock - Paper - Scissors
                  **************************



1
2
3

Press any key to continue . . .
```

since in the first case the user (first option) WON (remember `const int WIN = 1…`), in the second case the user LOST, and in the third case the user TIED.

<u>Show the instructor or TA your solution (STEP 4)</u>

When you are convinced this function is working, uncomment the five lines above, and delete any test cases like those given above that you may have created. Compile and run the program and make sure it is still functioning.

Step 5: Write `void displayWinnerMessage (char userSelection, char computerSelection)`

Your turn! Write this function, comment out the appropriate lines in main, and make particular function calls to this function to verify that it is working properly

<u>Show the instructor or TA your solution (Final Program)</u>

Optional Challenge:

1. Modify your program so that the user will continue playing the game until they select Q to quit the game.
2. Display a tally of number of WINS, LOSSES, or DRAWS.
3. Display the largest number of consecutive wins.

1) Your program is to compile without any errors or warnings.

2) Do not use any magic constants in your program. Define your constants before defining the rest of your program's variables.

Once your project is complete, place your solution into the CS150-03 Drop folder on Grace. Your solution is to have ALL previous projects completely working and correct.