

CSCI 1300 CS1: Starting Computing
Naidu/Correll/Yeh/Hoefer- Fall 2021
Project 1: Sequence of Cards
Due: Sunday, October 3rd, by 6pm

OBJECTIVES

- Understand and work with functions and strings
- Be able to test functions

You can find **Project1 background** on Canvas.

DO NOT USE GLOBAL VARIABLES IN YOUR SOLUTION. USE OF GLOBAL VARIABLES WILL RESULT IN A 0 ON THE PROJECT.

SUBMISSIONS

- Honor code questions (mcq). There are a few multiple-choice questions about the honor code. Don't forget to complete it!
- **C++ files.** All files should be named as specified in each question, and they should compile and run on Coderunner (pass all test cases) to earn full points. TAs will be grading style of your code and comments. Please see the style guide on Canvas and the summary note on Canvas. At the top of each file, write your name in the following format. If possible, do not use cin in your submitted code files, but rather directly call the functions you wrote.

```
// CSCI1300 Fall 2021
// Author: Punith Sandhu
// Recitation: 123 - TA name
// Project 1 - Problem # ...

#include <iostream>
using namespace std;

/*
 * This function converts a temperature in fahrenheit to celsius
 * and prints the equivalence.
 * Parameters: fahrenheit - degrees fahrenheit
 * Return: equivalent temperature in celsius
 */
double fahrenheit_to_celsius(double fahrenheit) {
    double celsius = (fahrenheit - 32) * (5/9.0);
```

```

        return celsius;
    }

    int main()
    {
        double fahrenheit1 = 32;
        double fahrenheit2 = 100;
        double fahrenheit3 = -30;
        cout << fahrenheit1 << "F is " <<
        fahrenheit_to_celsius(fahrenheit1) << " in celsius." << endl;
        cout << fahrenheit2 << "F is " <<
        fahrenheit_to_celsius(fahrenheit2) << " in celsius." << endl;
        cout << fahrenheit3 << "F is " <<
        fahrenheit_to_celsius(fahrenheit3) << " in celsius." << endl;

        return 0;
    }

```

For each question, create a program that calls the function in the main. When you are finished with all the questions, zip all files. Submit the zip file under the assignment project 1 zip file submission on Canvas.

- Code runner. Your program will be graded by the code runner. You can modify your code and re-submit (press Check again) as many times as you need to, up until the assignment due date.

INTERVIEW GRADING

Sign up for the interview grading slot on Canvas. You must sign up by September 29th. If you don't sign-up and you miss your interview, then the points for Coderunner will be replaced with 0. The schedulers for interview grading will be available before the deadline of this project (We will let you know when it opens).

START WITH THESE PROBLEMS

The following three questions are based on Strings. The questions are a good starting point and will help you get used to writing functions and to working with strings.

Question 1: **reverse()** (2 points)

Write a function named `reverse()` that takes a number (as a string) as a function parameter and prints that number in reverse.

Function Specifications:

- **Name:** `reverse()`
- **Parameters (Please Follow the same Order):**
 - `number (string)` - Number to be reversed
- **Return Value:** None
- **Example Function Call:** `reverse("1234567890");`
- The function should print the number in reverse.
- The function should receive the user input as the function parameter.

Sample Run 1 (**Text in Bold is User Input to be passed as Function Parameter**)

```
reverse("1234567890")
0987654321
```

Sample Run 2 (**Text in Bold is User Input to be passed as Function Parameter**)

```
reverse("04022021")
12022040
```

The file should be named as **`reverse.cpp`**. Once you have tested your code on VS Code, then head over to coderunner on Canvas and paste **only your function** in the answer box!

Question 2: `isConsonant()` (2 points)

Write a function **`isConsonant`** that accepts a character as an argument and returns TRUE if the character passed into the function is a Consonant, and FALSE otherwise. Consonant are the characters 'b', 'c', 'd', 'f', 'g', 'h', 'j', 'k', 'l', 'm', 'n', 'p', 'q', 'r', 's', 't', 'v', 'w', 'x', and 'z', and their uppercase variants.

Function Specifications:

- **Name:** `isConsonant()`
- **Parameters (Please Follow the same Order):**
 - `letter (char)` - Character to be checked
- **Return Value:** `true` or `false` (`bool`)
- **Example Function Call:** `isConsonant('a');`
- The function should be able to identify Consonant regardless of their case, i.e., even if they are in lowercase or uppercase. Do not use `islower()` or the `isupper()` functions.

Sample Run 1 (**Text in Bold is User Input to be passed as Function Parameter**)

```
bool consonantCheck = isConsonant('I');
cout << consonantCheck;
0
```

Sample Run 2 (**Text in Bold is User Input to be passed as Function Parameter**)

```
bool consonantCheck = isConsonant('k');
cout << consonantCheck;
1
```

The file should be named as **isConsonant.cpp**. Once you have tested your code on VS Code, then head over to coderunner on Canvas and paste **only your function** in the answer box!

Question 3: **numConsonant()** (3 points)

Write a function named `numConsonant()`. The function should accept one parameter of type string and it should return how many consonants are in the string. You should use the helper function `isConsonant()`, developed in the previous question.

Function Specifications:

- **Name:** `numConsonant()`
- **Parameters (Please Follow the same Order):**
 - `sentence (string)` - String to be examined for number of Consonants
- **Return Value:** Number of Consonants in the String - `num_Consonant (int)`
- **Example Function Call:** `numConsonant("Computer Science");`

Sample Run 1 (Text in Bold is User Input to be passed as Function Parameter)

```
int numOfConsonants = numConsonant("CSCI 1300: Starting Computing
1");
cout << numOfConsonants;
15
```

Sample Run 2 (Text in Bold is User Input to be passed as Function Parameter)

```
int numOfConsonants = numConsonant("Can the lockdown end already?");
cout << numOfConsonants;
14
```





















































The file should be named as **numConsonant.cpp**. Once you have tested your code on VS Code, then head over to coderunner on Canvas and paste **only your function** in the answer box!

DECK OF CARDS

The standard 52-card deck of French-suited playing cards is the most common pack of playing cards used today. A standard 52-card pack comprises 13 ranks in each of the four French suits: Clubs (♣), Diamonds (♦), Hearts (♥) and Spades (♠), with reversible (double-headed) court cards (face cards). Each suit includes an Ace, a King, Queen and Jack, each depicted

alongside a symbol of its suit; and numerals or pip cards from the Deuce (Two) to the Ten, with each card depicting that many symbols (pips) of its suit.

Example set of 52 playing cards; 13 of each suit: clubs, diamonds, hearts, and spades

	Ace	2	3	4	5	6	7	8	9	10	Jack	Queen	King
Clubs													
Diamonds													
Hearts													
Spades													

[Ref: Standard 52-Card Deck](#)

SEQUENCE OF CARDS

John, Mike and Suzy head to the Carnival at Zootopia. They come across a stall named “Can You Pick ‘em Right?!”. The stall owner Mr X greets them and invites them to play his game and tells them that the winner will get a coupon worth \$50 to spend in the Carnival. The three friends get excited and decide to play the game.

Mr X starts to explain the game to the participants. The ultimate aim of the game is for the players to create a sequence by picking a set of cards which is **similar** to the sequence picked by the host, i.e., Mr X. He says that every player is free to choose up to a maximum of 10 cards to form their sequence.

A sequence is formed by the cards picked by the players. The players select a subsequence of cards from their hand to compare against the golden sequence. The players cannot rearrange the cards, but can select a continuous subset that is the same length as the golden sequence. The game is played with *multiple decks* of cards. We know that in a deck of cards there are four suits: Spades (S), Hearts (H), Diamonds (D) and Clubs (C). And there are 13 ranks in each suit: Ace, 2-10, Jack, Queen and King. For the purpose of creating the sequence Ace is treated as the character A, Ten as T, Jack as J, Queen as Q and King as K.

Formation of Sequences:

- First, the card’s suit is taken and the letter representing that suit is added to the sequence.
- Next, the card’s rank is taken and appended to the sequence.
- Similarly, the process is repeated for all the cards picked by the player in the order that they pick the cards.

Let us assume that John has picked the following cards: **Ace of Diamonds, 5 of Hearts, Ten of Spades, Queen of Clubs** and the **6 of Hearts** in the same order as specified. Therefore the sequence generated by John is as follows:

DAH5STCQH6

Each player picks a maximum of 10 cards from which they can create their sequence. Once all the players have picked their cards, Mr X chooses his sequence of cards (without any knowledge of the cards picked by the players), which is going to be the **Golden Sequence** of cards. Mr. X can choose any number of cards less than or equal to 10. Let us assume that Mr X picks the following cards: **Ace of Diamonds, 4 of Hearts, Ten of Spades, 8 of Hearts** and **6 of Clubs**. So the Golden Sequence is as follows:

DAH4STH8C6

Once the Golden Sequence is found, a Likeness Score between the sequences of the players and the Golden Sequence is found. And finally, the player with the highest Likeness Score is declared the winner.

Rules for Likeness Score:

While finding the Likeness Score, more priority is given to the *Suit* of the cards rather than the Rank of the card. Rules include:

- First, the number of cards under consideration in the player's sequence and the Golden Sequence must be the same. So if the player has picked 10 cards, while the Golden Sequence has only 5 cards, then sub-sequences of 5 cards from the original sequence must be obtained. Something like this:

Player's Sub-Sequences To be Considered	Golden Sequence
HAD2S3C4H5D6S7C8H9CT	H6C8SJDKS3
HAD2S3C4H5D6S7C8H9CT	
HAD2S3C4H5D6S7C8H9CT	
HAD2S3C4H5D6S7C8H9CT	
HAD2S3C4H5D6S7C8H9CT	
HAD2S3C4H5D6S7C8H9CT	

- When the two sequences to be considered are available,

- The suit of the first card in both sequences are taken into account. If the suit is the same, then we keep track of those positions where the suit of the card is the same.
- Next, **if and only if** the suit of the cards under consideration is the same, we look at the rank of the card. If the rank of the cards are also the same, a bonus point of 1 (for every such match) is added in the calculation of the Likeness Score.
- Finally, once all the cards have been compared, the overall Likeness Score is calculated as follows:

$$\begin{aligned} \text{Likeness Score} &= \\ &= (\text{Number of Cards where the Suit is same} / \text{Number of Cards}) \\ &+ \\ &+ (1 * \text{Number of Cards with Same Suit and Same Rank}) \end{aligned}$$

Example: Let us look at John's sequence and the Golden Sequence

Cards

	#1	#2	#3	#4	#5
Player Sequence/Sub-Sequence	D A	H 5	S T	C Q	H 6
	+1	+0	+1	X -	X -
Golden Sequence	D A	H 4	S T	H 8	C 6

Not Considered as suit is different; leading to no bonus although rank is same

Considered; +1 bonus added as suit and rank are same

Considered; no bonus although same suit

Not Considered as suit is different; leading to no bonus

Therefore:

$$\text{Likeness Score} = \frac{3}{5} + 2 = 2.6$$

Thus, the Range of the Likeness Scores is: [0, 1 + Number of Cards in Sequences]

In this project (Questions 4-7) you will be creating an interactive program in C++ to allow users to play this game by implementing the following features:

Q4: calcLikenessScore() - A function to calculate the likeness between two sequences of equal length

Q5: `bestLikenessScore()` - A function to calculate the best Likeness Score among all the sub-sequences in a sequence and a given Golden Sequence

Q6: `findWinner()` - A function to find the winner among 3 players whose sequence has the best Likeness Score with the Golden Sequence.

Q7: Putting it All Together - Write a `main()` function that is going to call all the three functions created above to make the players play the game.

You're welcome to write additional helper functions as you need. Write your code and test each function on your VS Code. Then, once finished, you can submit it on Canvas coderunner to make sure it's fully functional. All the functions and the main function should be in one file, `SequenceOfCards.cpp`.

Question 4: `calcLikenessScore()` (8 points)

Write a function called `calcLikenessScore()` that is going to find the Likeness Score (as described earlier) between two sequences of equal length.

Function Specifications:

- **Name:** `calcLikenessScore()`
- **Parameters (Please Follow the same Order):**
 - `seq1` (string) - The First Sequence
 - `seq2` (string) - The Second Sequence
- **Return Value:** The Likeness Score - `likeness_score` (double)
- The parameters `seq1` and `seq2` should be of the same length for the function to calculate the Likeness Score. Else the Likeness Score returned should be -1.
- The function should not print anything.

Examples: Parameters and their expected Likeness Score which the function should return

<code>seq1</code>	<code>seq2</code>	<code>likeness_score</code>
S7H8CJD9HA	S7H8CJD9HA	6
C4DTSK	C4DJSK	3
HQDASJ	DAD8SJ	1.67
HJDKC3	C3HJDK	0
D7H2S4	H5DTS8C5	-1

The file should be named as `calcLikenessScore.cpp`. Once you have tested your code on VS Code, then head over to Coderunner on Canvas and paste **only your function** in the answer box!

Hint: Remember when you write a for loop, you can increment the value of the counter variable by any value and not just 1.

Question 5: **bestLikenessScore()** (8 points)

Write a function called `bestLikenessScore()` that is going to find the best Likeness Score between all the subsequences of a sequence (whose length is greater than or equal to the Golden Sequence) and the Golden Sequence. This function finds the maximum possible score a player could have.

Function Specifications:

- **Name:** `bestLikenessScore()`
- **Parameters (Please Follow the same Order):**
 - `seq1` (string) - The Player's Sequence (length greater than or equal to Golden Sequence)
 - `gold_seq` (string) - The Golden Sequence
- **Return Value:** Best Likeness Score - `best_likeness_score` (double)
- Length of `seq1` should be greater than or equal to `gold_seq` for the function to calculate the Best Likeness Score. If not, return -1.
- Compare the sub-sequences from the player's sequence to the Golden Sequence to find the Best Likeness Score among all the sub-sequences.
- Use the previous `calcLikenessScore()` function from Question 4 to find the Likeness Score once the sub-sequence is obtained.
- The function should not print anything.

Example: Expected output from a Longer Player's Sequence and the Golden Sequence

seq1	seq2	likeness_score
S7H8SJD9 H8CJD9	H8C6D6	0
S7 H8SJD9 H8CJD9		1.67
S7H8 SJD9 H8CJD9		0
S7H8SJD 9H8CJD 9		0
S7H8SJD9 H8CJD 9		2

The function should calculate the Likeness Score between all the subsequences and the Golden Sequence and should return only the Best Likeness Score as indicated. In the example above, the last subsequence has the highest score, 2, and therefore 2 should be the return value.

The file should be named as **bestLikenessScore.cpp**. Once you have tested your code on VS Code, then head over to coderunner on Canvas and paste **only your function** in the answer box!

Question 6: **findWinner()** (7 points)

Write a function called `findWinner()` that is going to find the Winner among 3 players and display the winner in the console. The players have the option to pick a maximum of 10 cards and form their sequences. It is not necessary for the players to choose the same number of cards. These sequences are passed into the function as `seq1`, `seq2`, `seq3`.

The length of the Golden Sequence will always be lesser than or equal to the length of the smallest sequence among the 3 players.

Function Specifications:

- **Name:** `findWinner()`
- **Parameters (Please Follow the same Order):**
 - `seq1` (string) - Player 1 Sequence
 - `seq2` (string) - Player 2 Sequence
 - `seq3` (string) - Player 3 Sequence
 - `gold_seq` (string) - The Golden Sequence
- **Return Value:** No Return Value.
- **Output:**
 - **Remember, that in order for a player to win his/her score should be greater than 0.** If all three players have a score of 0 then there is no winner and the function should print **"Better luck next time everyone!"**
 - If all the three players have the same score, then the function should print **"Congratulations everyone! You have all won!"**
 - If there are two players with the same best score, the function should print **"Congratulations Players # and #! You have won!"** where the #s represent the winning players.
 - If there is a clear winner, then the function should print **"Congratulations Player #! You have won!"**, where # is either 1, 2 or 3.
- Remember the sequences `seq1`, `seq2` and `seq3` of the players need not be of the same length, i.e., they need not pick the same number of cards.
- Use the previous `bestLikenessScore()` function from Question 5 to find the Best Likeness Score in the entire sequence of the player.

Examples:

Case1: All players with a score of 0:

Player #	seq#	gold_seq	Best Likeness Score
1	CAH7S5CJCK	D4D5D6	0
2	S7H9SQCA		0

3	H2S6H7CTS9HK		0
<p style="text-align: center;">EXPECTED OUTPUT</p> <p style="text-align: center;">Better luck next time everyone!</p>			

Case2: All players with same Best Likeness Score:

Player #	seq#	gold_seq	Best Likeness Score
1	DAH7S5CJ	D4D5D6	0.33
2	S7H9DQCA		0.33
3	H2S6CTD7		0.33
EXPECTED OUTPUT			
Congratulations everyone! You have all won!			

Case3: Two players with same Best Likeness Score:

Player #	seq#	gold_seq	Best Likeness Score
1	DAH7S5CJ	DTD2D3	0.33
2	S7D2DQCAH7		1.67
3	H2DTCTD7S8C5		1.67
EXPECTED OUTPUT			
Congratulations Players 2 and 3! You have won!			

Case4: Clear Winner with the Best Likeness Score:

Player #	seq#	gold_seq	Best Likeness Score
1	H2DTHTSAS7CA	DTH7SA	3
2	D7H2SQCAH7		1
3	CJDAH7C5		1.67
EXPECTED OUTPUT			
Congratulations Player 1! You have won!			

The file should be named as **findWinner.cpp**. Once you have tested your code on VS Code, then head over to coderunner on Canvas and paste **only your function** in the answer box!

Question 7: Putting it all Together using `main()` (5 points)

In this question, you will be writing a `main()` driver function through which you will give the end user the option to choose from the following options:

- Calculate the Likeness Score between two sequences of equal length
- Find the Best Likeness Score between two sequences of unequal length
- Find the Winner among the sequences of 3 players and the Golden Sequence
- Exit

The menu will run on a loop, continually offering the user four options until they opt to quit. You need to fill in the code for each of the options. Be sure to use the functions you wrote in questions 4-6.

Below is an Sample Run of the main program depicting different cases:

(Text in Bold represents User Input)

```
----Menu!----
Choice 1: Calculate the Likeness Score between 2 strings of equal
length.
Choice 2: Calculate the Best Likeness Score between 2 sequences of
possibly different length.
Choice 3: Find winner among sequences of 3 players and a Golden
Sequence.
Choice 4: Exit.
Enter your choice (1-4):
1
Enter Sequence 1
HQDASJ
Enter Sequence 2
DAD8SJ
Likeness Score: 1.66667

----Menu!----
Choice 1: Calculate the Likeness Score between 2 strings of equal
length.
Choice 2: Calculate the Best Likeness Score between 2 sequences of
possibly different length.
Choice 3: Find winner among sequences of 3 players and a Golden
Sequence.
Choice 4: Exit.
Enter your choice (1-4):
2
Enter Sequence 1
S7H8SJD9H8CJD9
Enter Sequence 2
H8C6D6
```

Best Likeness Score: 2

----Menu!----

Choice 1: Calculate the Likeness Score between 2 strings of equal length.

Choice 2: Calculate the Best Likeness Score between 2 sequences of possibly different length.

Choice 3: Find winner among sequences of 3 players and a Golden Sequence.

Choice 4: Exit.

Enter your choice (1-4):

3

Enter Sequence of Player 1

CAH7S5CJCK

Enter Sequence of Player 2

S7H9SQCA

Enter Sequence of Player 3

H2S6H7CTS9HK

Enter Golden sequence

D4D5D6

Better luck next time everyone!

----Menu!----

Choice 1: Calculate the Likeness Score between 2 strings of equal length.

Choice 2: Calculate the Best Likeness Score between 2 sequences of possibly different length.

Choice 3: Find winner among sequences of 3 players and a Golden Sequence.

Choice 4: Exit.

Enter your choice (1-4):

3

Enter Sequence of Player 1

DAH7S5CJ

Enter Sequence of Player 2

S7H9DQCA

Enter Sequence of Player 3

H2S6CTD7

Enter Golden sequence

D4D5D6

Congratulations everyone! You have all won!

----Menu!----

Choice 1: Calculate the Likeness Score between 2 strings of equal length.

Choice 2: Calculate the Best Likeness Score between 2 sequences of possibly different length.

Choice 3: Find winner among sequences of 3 players and a Golden Sequence.

Choice 4: Exit.

Enter your choice (1-4):

3

Enter Sequence of Player 1

DAH7S5CJ

Enter Sequence of Player 2

S7D2DQCAH7

Enter Sequence of Player 3

H2DTCTD7S8C5

Enter Golden sequence

DTD2D3

Congratulations Players 2 and 3! You have won!

----Menu!----

Choice 1: Calculate the Likeness Score between 2 strings of equal length.

Choice 2: Calculate the Best Likeness Score between 2 sequences of possibly different length.

Choice 3: Find winner among sequences of 3 players and a Golden Sequence.

Choice 4: Exit.

Enter your choice (1-4):

3

Enter Sequence of Player 1

H2DTHTSAS7CA

Enter Sequence of Player 2

D7H2SQCAH7

Enter Sequence of Player 3

CJDAH7C5

Enter Golden sequence

DTH7SA

Congratulations Player 1! You have won!

----Menu!----

Choice 1: Calculate the Likeness Score between 2 strings of equal length.

Choice 2: Calculate the Best Likeness Score between 2 sequences of possibly different length.

Choice 3: Find winner among sequences of 3 players and a Golden Sequence.

Choice 4: Exit.

Enter your choice (1-4):

1

Enter Sequence 1

D7SAH9

```

Enter Sequence 2
CTH9
Invalid input. Sequences of different length.

----Menu!----
Choice 1: Calculate the Likeness Score between 2 strings of equal
length.
Choice 2: Calculate the Best Likeness Score between 2 sequences of
possibly different length.
Choice 3: Find winner among sequences of 3 players and a Golden
Sequence.
Choice 4: Exit.
Enter your choice (1-4):
5
Invalid Input. Choices between 1-4.

----Menu!----
Choice 1: Calculate the Likeness Score between 2 strings of equal
length.
Choice 2: Calculate the Best Likeness Score between 2 sequences of
possibly different length.
Choice 3: Find winner among sequences of 3 players and a Golden
Sequence.
Choice 4: Exit.
Enter your choice (1-4):
4
Exiting.

```

The file should be named as **sequenceOfCards.cpp**. Once you have tested your code on VS Code, then head over to coderunner on Canvas and paste the **entire program** (functions from Questions 4-6 and the main function from this Question 7) into the answer box!

Extra Credit Question 1: **Creating a Random Sequence** (3 points)

*****Note random numbers can be generated using the rand() function. See video tutorials [here](#) for more information.*****

(https://www.youtube.com/playlist?list=PL8ID6FG_UGW-MZtnkqzWcN5VudlForPkb)

The challenge is to randomly create a sequence of the cards when the number of cards making up the sequence is given. Remember, that a card is made up of two characters, one representing the **Suit** (S, C, D, H) and the other representing the **Rank** (A, 2-9, T, J, Q, K). And combining such cards will form a sequence!

Example:

Number of Cards = 3 Number of Cards = 5	SAD6C9 C3C9D6SAHT
--	----------------------

(Remember the above examples are just for reference. Your code need not output the same sequences for inputs 3 and 5.)

Function Specifications:

- **Name:** `generateSequence()`
- **Parameters (Please Follow the same Order):**
 - `numOfCards (int)` - Number of Cards forming the sequence
- **Return Value:** Random Sequence of Cards - `generated_seq (string)`

The file should be named as **`generateSequence.cpp`**. Once you have tested your code on VS Code, then head over to coderunner on Canvas and paste **only your function** in the answer box!

Extra Credit Question 2: **Shuffling a Sequence** (5 points)

Consider you have a sequence as specified in the previous questions. The challenge is to shuffle the cards in random order. Remember, the *cards* have to be shuffled in the sequence. So, the suit and rank of any card should always be together. In this question you will write a function named `cardsShuffle()` that takes the sequence of the cards as the parameter and returns the shuffled sequence.

Example:

Original Sequence:

SAD6C9HTC3

Shuffled Sequence:

C3C9D6SAHT

Function Specifications:

- **Name:** `cardsShuffle()`
- **Parameters (Please Follow the same Order):**
 - `orig_seq (string)` - The Sequence to be shuffled
- **Return Value:** Shuffled Sequence - `shuffled_seq (string)`

The file should be named as **`shuffleSequence.cpp`**. Once you have tested your code on VS Code, then head over to coderunner on Canvas and paste **only your function** in the answer box!

PROJECT 1 CHECKLIST

Here is a checklist for submitting the assignment:

1. Complete the Honor code questions
2. Complete the code Project 1 CodeRunner
3. Submit one zip file to project 1 zip file submission. The zip file should be named, `project1_lastname.zip`. It should have the following 9 files (last two are extra credit):
 - **`reverse.cpp`**

- `isVowel.cpp`
 - `numVowels.cpp`
 - `calcLikenessScore.cpp`
 - `bestLikenessScore.cpp`
 - `findWinner.cpp`
 - `sequenceOfCards.cpp`
 - `generateSequence.cpp`
 - `shuffleSequence.cpp`
4. Sign up for the Interview Grading Slot (15 minutes) on Canvas. If you don't sign-up and you miss your interview, then no points will be awarded for the project. The schedulers for interview grading will be available before the deadline of this project (We will let you know when it opens).

During the interview grading, your TAs will be checking style and comments of your code and test cases. They will also ask about your implementations and conceptual questions.

PROJECT 1 SUMMARY

Criteria	Pts
Code Runner (Questions 1-7)	35
Interview Grading	15
Recitation attendance (if missed)	(-15)
TOTAL Extra Credit	50 +8