**CS 201R**

**Problem Solving & Programming II**

**Program 4 – Games Due \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Key Learning Objectives:**

* Functions (pass by value vs reference)
* Function files (.h & .cpp)
* Vector (of struct)
* Looping, Branching, Print formatting
* File input
* String manipulation & special characters (extra credit)

**Assignment Problem:**

You have been so bored in class that you decided to create a series of games for entertainment.

Text

Description automatically generated

To begin, ask the user which of the games they would like to play.

Then, the player gets to play the game chosen and when done, can choose another game to play.

**Blackjack:**

The game opens by creating a deck of 52 cards and 2 empty hands (the player and computer hands).

Initial Deal: Deal 2 cards to each hand (the player and computer). Take the top card from the deck and add it to the player hand, remove that card from the deck. Then deal the next top card from the deck and add to the computer hand and remove that card from the deck and repeat. Thus, dealing 4 cards.

Player turn: Display the players cards and their hand total and ask if they would like another card. If so, take the top card from the deck and add it to the player hand, remove that card from the deck. Calculate the new player total and display that value and repeat, until the player does not want another card. If the player’s hand total exceeds 21, they automatically lose.

Dealer turn: The computer **must** deal a card if the sum of the cards in their hand is <= 17. If the computer’s hand total exceeds 21, it automatically loses.

Evaluate who won the game (whoever has the highest value that did not exceed 21). If the player and the dealer end up with the same hand total, then this is called a PUSH

NOTES:

1. Aces in blackjack can count for 1 or 11. Initially they should count as 11, but if the player goes over 21, change the value of the ace to 1.
2. If the player would like to play again, a new deck should be created
3. Display of the special characters and updating the color of the console is not difficult, but counts as extra credit. You may choose to write out the suit of the card instead.

**Sample run:**

Text

Description automatically generated

Text

Description automatically generated

**Craps:**

The simplified rules of this game are based on the rolling of 2 six sided die.

if 7 or 11 – you win

if 2,3,12 – you lose

all others establish ‘point’

keep rolling until :

point – you win

7 – you lose

These are the same rules established when done in class during week 1. There is an additional twist to allow the user to bet on the outcome of their craps game. The player gets a starting bet of $50 and can wager a bet based on this maximum amount. The player cannot bet more than his/her worth, and once his/her worth is $0, the game must end. Otherwise, you can continue to prompt the player to go again. When the player chooses to quit (before running out of money), please display the player’s worth.

**Sample output:**

A picture containing text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

**Hangman:**

The pseudocde done in class is sufficient for this game. You will read a list of words, (see the list here: [**words.txt**](https://umkc.box.com/s/m2iej34ldg8bwjvumgpdt93naillo6d8)) store them in a vector, and then randomly choose a word from the list. You will need to create a ‘guessWord’ that contains either ? or \_ for each character in the chosen word. The user will guess the letter until they run out of their 6 guesses the guess word equals the chosen word.

**Sample Output**

Text

Description automatically generated

Text

Description automatically generated

These games could mostly be written from pseudocode written in class. Your final game can be either Tic Tac Toe **or** War.

Text

Description automatically generated

**TicTacToe**

For this game, you start by ‘flipping a coin’ to decide if the player or computer will go first.

The board will be displayed, and if the computer goes first, it will fill in a ‘O’ on the board, display the board, then allow the user to choose the position they prefer from 1-9 and when chosen, you will place an ‘X’ in that position and continue alternating turns between the computer and the player.

The computer will follow logical rules for play as follows:

1. machineRule1: IF Machine or Player can Win [ IF anyone needs 1 place to Win ] -> The Machine Fill That Place.
2. machineRule2: IF the Center is open -> Fill the center.
3. machineRule3: IF Player Fill a Corner -> Fill the opposite corner.
4. machineRule4: IF none of the previous rules were used, fill in any a corner.
5. machineRule5: IF none of the previous rules were used, fill the first available place.

**War**

The basic goal of this game is for the player or the computer to get all the cards.

To start:

1. Create the deck, shuffle the deck and deal all of the cards to the player and computer hands.
2. Each player will show their ‘top’ card.
   1. Compare players’ card value (Do not consider the color and pattern)
   2. The player that has the higher value revealed card takes both cards on the desk and place those cards facing down at the bottom of his/her own deck.
   3. If the computer & the player revealed cards with same value, this starts ‘war’.
3. War play
   1. When the war begins, both players place their next 3 cards from the top of his/her own deck in the ‘hold’ hand.
   2. Then, each player will show their ‘top’ card and the comparison is done to win or lose.
   3. The player with the highest value card wins all the cards (the initial tied cards, the 6 cards placed ‘face down’ and the last two cards shown, for a total of 10 cards.
   4. If there is a tie, War is initiated again.
4. The players continue to reveal the top card of their deck until a winner is declared. The player to reach a total of 52 cards is the winner. Because the program will run excessively long at times, I suggest you declare a winner when one player has less than 10 cards (the opposing player wins). Otherwise, if one player is not able to complete the “war” (do not have enough cards to complete the “war”), this player loses the game, and the other player wins

**Sample output:**

A picture containing arrow

Description automatically generated

Text

Description automatically generated

**Bonus:**

* Display of the special characters for the card deal and updating the color of the console for Blackjack (and War if you choose to do this). 15 points
* You may choose to do both Tic Tac Toe AND War. 20 points

**Submission:**

* NOTE: there is probably code for these games available ‘out there’, but which may not meet the criteria of the games as described above. If you choose to use available code, you MUST site your source and you MUST make the proper adjustments to meet the requirements given here.
* Complete [THIS FORM](https://forms.gle/MuegLwwo8xCJRaYo8) for turn in. You will need to include:
  + Your pseudocode. This can be a link to a google document, an image, or any site used to make your pseudocode. Make sure to share this with [gladbachj@umkc.edu](mailto:gladbachj@umkc.edu) and <stgdcg@umsystem.edu>.
  + You will complete your code in Visual Studio. Please then copy your code to your repl.it space, ensure the program is running as expected, and supply the repl.it link in this form

**Rubric:**

This is the rubric used for grading [CS201 Program 4 - Games](https://umkc.box.com/s/xpwmm37m3fs4x56fojahuokmiedr3nxe)