GAMES NAME: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**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)

|  |  |  |  |
| --- | --- | --- | --- |
| **OBJECTIVE** | **DESCRIPTION** | **RESULTS EXPECTED** | **POINTS\*** |
| **0. Pseudocode** | Pseudocode provided is clear, easy to follow and allows the program to follow program logic |  | **/20** |
| **1. Accuracy** | Menu is clearly and neatly formatted |  | **/5** |
|  | Error in provided if an incorrect choice is made |  | **/5** |
| **NOTE:**  each item listed as part of the expected results is worth 5 points unless otherwise noted. | **Blackjack** Functionality | * Player’s cards are displayed neatly & player’s total is correct * Prompt to continue dealing is provided * Updated player total is correct * Errors as expected if player (or computer) goes over 21 * Correctly reevaluates an ace when hand may exceed 21 * Displays ‘push’ if player and computer tie * Plays computer hand correctly | **/35** |
|  | **Craps** Functionality | * Allows user to place a bet * Bets exceeding the player’s worth are rejected * Bets are updated appropriately based on the win/loss * Follows all rules of craps * Correctly plays the ‘point’ logic | **/25** |
|  | **Hangman** Functionality | * Reads the word file and chooses a random word * Guess word is displayed with ? * Accurately updates guess word if letter guessed is in the word * Correctly tracks the number of attempts made for incorrect guesses * Ends the game by displaying win or loss and word chosen | **/25** |
| **Chose**  **Tic-Tac-Toe**  **OR**  **War** | **Tic-Tac-Toe** Functionality | * Coin flip for game start * Neatly displays board * Correctly fills move chosen by the player * Displays a warning if the position chosen is not open * Correctly follows logic for computer play   + Rule 1: fill win position   + Rule 2: fill center   + Rule 3: fill opposite corner   + Rule 4: fill corner   + Rule 5: first open space * Correctly displays the win/loss/draw (10 points) | **/55** |
|  | **War** Functionality | * Creates the deck & shuffles * Deals 26 cards to each player to start * Displays the accurate card count of the cards held by Player & Computer after each card is played * Displays the Player, Computer, & Hold hand neatly * Correctly plays the ‘top’ card for each player & modifies the player’s hand & card count * Correctly evaluates if the player or computer win or if it is war * Correctly modifies the hand based on the card value * Plays war accurately   + Display all cards in the ‘hold’ hand   + Deals 3 additional cards from the player and computer hand   + evaluates the next card played and accurately moves the cards in the hold hand to the winning player * Correctly concludes the game | **/55** |
| **2. Style** | Code is well organized  & easy to read | * Related code is put together * Indentions are properly used to improve readability * Variable names are meaningful | **/10** |
|  | Proper functions and function files are used | * .h and .cpp files are appropriately setup * pre & post comments for functions are included | **/10** |
|  | Code is well commented | * Heading information includes Name, Date & Program name * Brief explanations before blocks of related code | **/10** |
| **Bonus Feature(s)** | * 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) * You may choose to do both Tic Tac Toe AND War. |  | **+15**  **+20** |
| **TOTAL** | ADDITIONAL COMMENTS: |  | **/200** |

**POINTS\***

Points assigned are partially subjective, but you can expect points assigned as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | 2 - 3 | 4 | 5 (5 point scale) |
| 0 | 1-5 | 6 – 7 | 8-9 | 10 (10 point scale) |
| 0 | 1-10 | 12-14 | 15-19 | 20 (20 point scale) |
| No evidence of meeting this requirement | Evidence of requirement, but working at a substandard level  (code is present, but does not compile, for example) | Evidence of requirement, partially working  (code is present, some values work, but other values cause an abend for example) | Requirement is predominantly met, but has some small issues  (code is present and runs, but output is not completely correct, for example) | Requirement is fully met and functioning |