CS XXX – Rust

Fall 2025

PEX 1: BlackJack

Help Policy

AUTHORIZED RESOURCES: Any, except another cadet's assignment or published solutions to the assigned problem.

NOTE:

- Never copy another person's work and submit it as your own. Here are a few blatant examples of copying:
  - Making an electronic copy of another cadet's solution and then modifying it slightly to make it appear as your own work.
  - Reading a printout or other source of another cadet's work as you implement your solution.
  - Completing your entire solution by following explicit instructions from another cadet, while he/she refers to his/her own solution
- Do not jointly implement a solution.
- Helping your classmates learn and understand the homework concepts is encouraged, but extensive assistance should generally be provided by DFCS instructors. Only provide assistance up to your depth of understanding, beyond which assistance by more qualified individuals is more appropriate and will result in greater learning. If you have to look at your solution while giving help, you are most likely beyond your depth of understanding.
- Help your classmates maintain their integrity by never placing them in a compromising position. Do not give your solution to another cadet in any form (hard copy, soft copy, or verbal).
- You **may not** solve the entire PEX with ChatGPT or any other "generative AI" technology.
- You **may** ask the generative AI questions to fix your code, and you may copy and paste your code into the generative AI. You must document what you asked the generative AI (via hyperlink to the session transcript), provide a description of the error in your code, and describe how you were able to fix it.
- You **may** ask the generative AI questions about how to program a general concept related to the PEX
- **DFCS will recommend a grade of A for any cadet who egregiously violates this Help Policy or contributes to a violation by others. Allowing another cadet to see your assignment to help them will result in a zero on this assignment.**

Documentation Policy

- You must document all help received from sources other than your instructor or instructor-provided course materials (including your textbook).
- The documentation statement must explicitly describe WHAT assistance was provided, WHERE on the assignment the assistance was provided, and WHO provided the assistance.
- If no help was received on this assignment, the documentation statement must state "NONE."

- If you checked answers with anyone, you must document with whom on which problems. You must document whether or not you made any changes, and if you did make changes you must document the problems you changed and the reasons why.
- Vague documentation statements must be corrected before the assignment will be graded and will result in a grade deduction equal to 5% (ceiling) of the total possible points.

OBJECTIVES

- Understand and apply the following Rust concepts:
  - Matching
  - Enums
  - References
- Gain familiarity with Rust's unique syntax and rules

OVERVIEW

In this project, you will build a simple command line blackjack game in which the player plays against a simulated casino-style blackjack dealer. This will require implementing a control loop, a scoring mechanism, and implementing applicable data structures.

SUBMISSION INSTRUCTIONS

Submission via Github

NOTE: your documentation statement must be in your main.rs.

GENERAL REQUIREMENTS

- You must implement this program in Rust
- Your project should compile cleanly (i.e. free of any warnings or errors)
- Utilize Rust language features where reasonable and applicable
- Provide sufficient, relevant comments

PROGRAM REQUIREMENTS

- Program shall implement standard Blackjack Game against a simulated dealer

- Dealer shall use standard dealing rules (hit on 16, stand on 17)

- Program shall correctly score hands, including aces

- Program shall implement and use a Card Enum

- Program shall correctly utilize matching for the Card Enum

- Program shall gracefully handle invalid user input

- Program shall prompt user for a name and use it for relevant interactions

- Program shall display final result of the game
- Program shall display initial state of a hand
- Program shall display all actions by both user and dealer and updated game state

HINTS/PLAN OF ATTACK

Consider what attributes of a card are relevant in blackjack. Suites are not used, so there is no point in generating and tracking them for each card. Using the rand package is strongly advised for generating new cards. A match statement will also likely be helpful for this.

For handling aces, consider evaluating aces after the rest of the hand, because their value is determined by the rest of the hand. Additionally, consider the effect of multiple aces in a single hand and how you will determine how many will count for eleven and how many will count for one.

Guiding Questions:
What associated values are relevant for each type of card?
What is different between the player and dealer stages of the game?
What is the maximum number of aces which can possibly take a value of eleven?