Runtime Error

Necolai McIntosh

Overview

Premise

Runtime Error is a game about programming, bug testing, and infinite loops. The game centers around the creation of conditional statements using cards, and the goal is to find a way to make your turn last forever.

Setup

Materials

- 2-4 players
- Standard deck of 52 playing cards (without Jokers)

How to Set Up

Shuffle a standard deck of playing cards, place the deck near the center of the table, and deal 5 cards to each player. Choose a player to go first (preferably the player with the most programming experience), then play proceeds clockwise.

Gameplay

The object of *Runtime Error* is to initiate an infinite loop on your turn.

Each turn is made up of three phases:

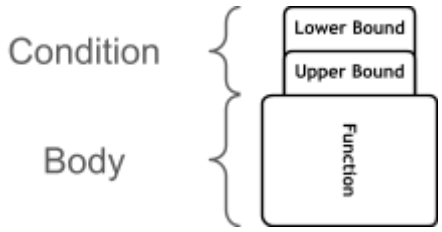
Scripting Phase:
Place three cards from your hand in front of you to create a statement. You may skip this phase if you want, but you can only create one statement per turn.

Testing Phase:
Play a card from your hand in the center of the table to use as the *input card*. If the value of the input card satisfies any of your statements' conditions, execute those statements' functions. You may skip this phase if you want, but you can only play one input card per turn unless stated otherwise.

End of Turn Phase:
Discard the current input card if there is one, then draw a card from the deck and add it to your hand.

Scripting Phase

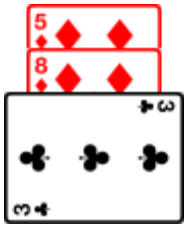
During the scripting phase, you may create one statement and place it in front of you. Each statement consists of three cards arranged like so:



The *condition* of a statement consists of two cards of the same suit, the *lower bound* and *upper bound*, overlapping downward. During the testing phase, the values of the lower and upper bounds dictate the minimum and maximum input card values that will satisfy the condition, where Aces, Jacks, Queens, and Kings represent the values 1, 11, 12, and 13 respectively.

The *body* of a statement consists of one card, the *function*, placed horizontally on top of the condition. During the testing phase, the suit of the function determines which action the player should perform if the input card satisfies the statement's condition.

For example, the following statement has a lower bound of 5, an upper bound of 8, and a body containing the Club function.



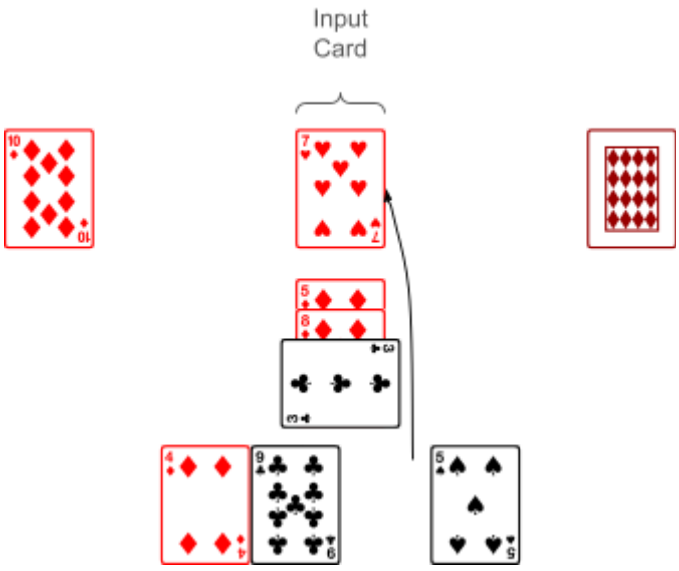
Statements must also abide by the following syntax rules in order to be considered valid:

- The value of the lower bound must be less than that of the upper bound.
- The lower bound and upper bound must have the same suit.
- You may not have two statements with functions of the same suit.

You may skip this phase if you want, but you can only create one statement per turn.

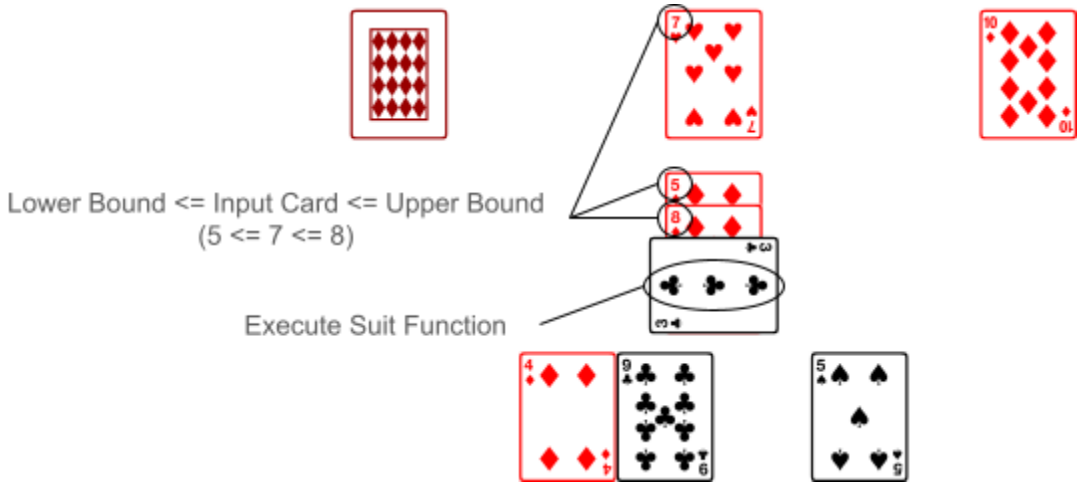
Testing Phase

To begin the testing phase, choose a card from your hand to use as the “input card” and place it in the center of the table.



Next, check the conditions of each of your statements to see which of them are satisfied by the value of the input card. If the input card’s value falls within the lower and upper bounds of one of your statements, that statement’s condition is satisfied.

In the example below, playing an input card with a value of 5, 6, 7, or 8 will satisfy this statement’s condition, causing the Club function to be executed.



If the input card satisfies a statement's condition, you must perform the action that corresponds with the suit of that statement's function.

- Diamond Function (♦): EDIT
    - Choose any card from one of your own statements or from one of your opponent's statements, discard it, and replace it with a card from your own hand face-down. The edited statement remains inactive until after the current input card has been discarded, at which point the edited card is flipped face-up.
        - You may not replace a card in order to create an invalid statement.
- Club Function (♣): LOAD
    - Draw an extra card from the top of the deck and add it to your hand.
- Heart Function (♥): PULL
    - Draw a card from the top of the discard pile and add it to your hand.
        - If there are no cards in the discard pile, ignore this statement.
- Spade Function (♠): LOOP
    - Discard the current input card, then play another input card from your hand and repeat the testing phase.
        - If there are no cards in your hand, ignore this statement.

If the input card satisfies more than one of your statements' conditions, you must execute their functions in the following order: Diamonds (♦), then Clubs (♣), then Hearts (♥), then Spades (♠).

If there are no more cards in the deck, shuffle the discard pile and use it as the new deck.

If, on your turn, there exists a series of actions that you can perform to make your turn to last forever, you win! You must be able to articulate, or at least demonstrate, why your turn will last forever in order to secure your victory.

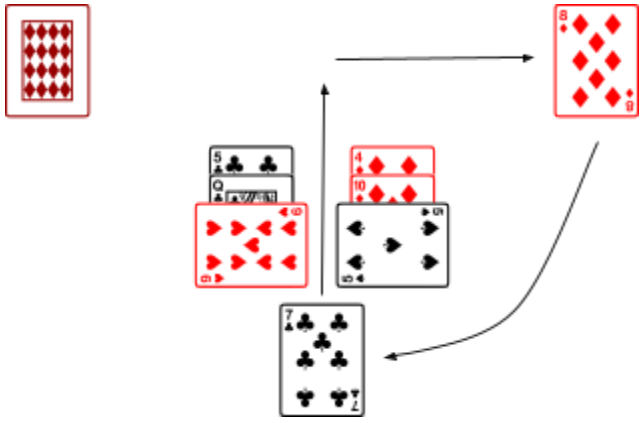
You may skip this phase if you want, but you can only play one input card per turn unless stated otherwise.

End of Turn Phase

Discard the current input card if there is one, then draw a card from the top of the deck and add it to your hand.

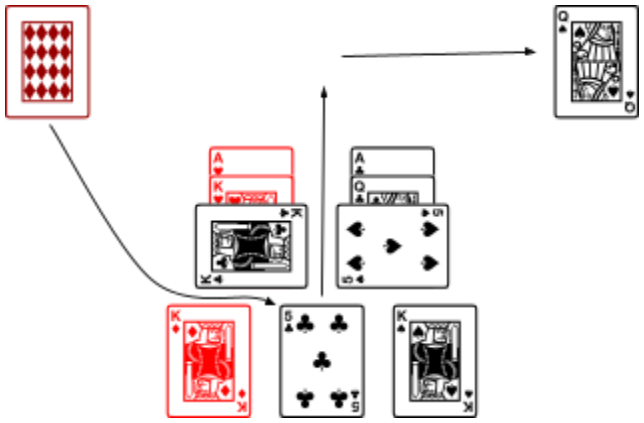
Win Condition Examples

Example 1: Hearts and Spades



1. The player plays the 7 of Clubs from their hand as the input card.
2. The leftmost statement executes the Heart function if the input card has a value from 5 to Queen. The input card's value of 7 satisfies this condition, so the Heart function is executed.
3. The player draws the 8 of Diamonds from the top of the discard pile and adds it to their hand.
4. The rightmost statement executes the Spade function if the input card has a value from 4 to 10. The input card's value of 7 satisfies this condition, so the Spade function is executed.
5. The player discards the input card, then plays the 8 of Diamonds from their hand as the new input card.
6. The leftmost statement executes the Heart function if the input card has a value from 5 to Queen. The input card's value of 8 satisfies this condition, so the Heart function is executed.
7. The player draws the 7 of Clubs from the top of the discard pile and adds it to their hand.
8. The rightmost statement executes the Spade function if the input card has a value from 4 to 10. The input card's value of 8 satisfies this condition, so the Spade function is executed.
9. The player discards the input card, then plays the 7 of Clubs from their hand as the new input card.
10. The player can repeat this process indefinitely, so they win.

Example 2: Clubs and Spades

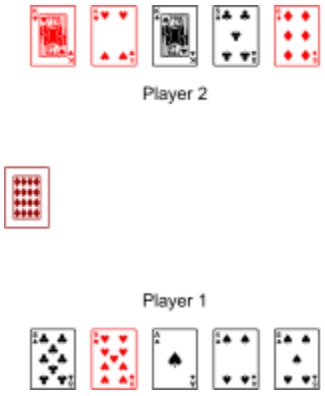


1. The player plays the 5 of Clubs from their hand as the input card.
2. The leftmost statement executes the Club function if the input card has a value from Ace to King. The input card's value of 5 satisfies this condition, so the Club function is executed.
3. The player draws a card from the top of the deck and adds it to their hand.
4. The rightmost statement executes the Spade function if the input card has a value from Ace to Queen. The input card's value of 5 satisfies this condition, so the Spade function is executed.
5. The player discards the input card, then plays the card that they just drew from their hand as the new input card.
6. The leftmost statement executes the Club function if the input card has a value from Ace to King. Any card will satisfy this condition, so the Club function will always be executed.
7. The player draws a card from the top of the deck and adds it to their hand.
8. The rightmost statement executes the Spade function if the input card has a value from Ace to Queen. Any card except for Kings will satisfy this condition, and since there are no Kings in the deck or the discard pile, the Spade function will always be executed.
9. The player discards the input card, then plays the card that they just drew from their hand as the new input card.
10. The player can repeat this process indefinitely, so they win.

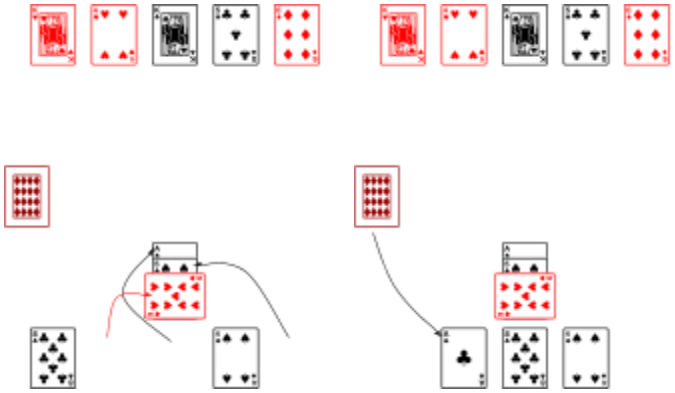
Sample Round

The section below shows the first round of play from a 2-player game of *Runtime Error*.

The game begins as both players are dealt the following cards:



Player 1 goes first, creating a statement with a lower bound of Ace, an upper bound of 5, and a Heart function in the body. Player 1 skips the testing phase and draws a card from the deck, ending their turn.



Player 2 goes next, creating a statement with a lower bound of 4, an upper bound of King, and a Diamond function in the body. Player 2 then plays an input card with a value of 5, which satisfies their statement's condition, executing the Diamond function. Player 2 uses the Diamond function to edit Player 1's statement, then discards the input card. Finally, Player 2 draws a card from the deck, ending their turn.