
# Card Game

**Joseph Gutierrez**
**CSC 5 45113**
**Dr. Lehr**

# How To Play

The rules are pretty simple. The cards are divided into 4 colors, Red Green Blue Yellow, and numbered 0-9. There are special action cards such as Draw Two and Skip Turn.

## Goal

The goal is for the player to reach zero cards in their hands. Whoever loses their whole hand first wins.

## Gameplay

- A random card is used to start. The player must match the card on top of the pile with one of either the same color or number from their hand.
- Placing a Draw Two card means the next player has to draw two cards from the deck.
- Placing a Skip Turn card means the next player's turn is skipped.

If the card in a player's hand does not match, they must draw from the deck until a card of either the same color or number as the top card is drawn and placed.

# My Gameplay Approach

Your cards are printed with the corresponding card number on top. You input the number that corresponds its card. If the card matches in either color or number to the top card, the card will be placed and the turn will end. The CPU will then play its turn.

Additionally, to draw cards at your own will, you can input ‘d’ to draw from the top of the deck.

# Cross Reference for Project 1

## You are to fill-in with where located in code

| Chapter | Section | Topic | Where Line #''s | Pts | Notes |
|---|---|---|---|---|---|
| 2 | 2 | cout | | | |
| | 3 | libraries | | 8 | iostream, iomanip, cmath, cstdlib, fstream, string, ctime |
| | 4 | variables/literals | | | No variables in global area, failed project! |
| | 5 | Identifiers | | | |
| | 6 | Integers | | 3 | |
| | 7 | Characters | | 3 | |
| | 8 | Strings | | 3 | |
| | 9 | Floats  No Doubles | | 3 | Using doubles will fail the project, floats OK! |
| | 10 | Bools | | 4 | |
| | 11 | Sizeof ***** | | | |
| | 12 | Variables 7 characters or less | | | All variables <= 7 characters |
| | 13 | Scope *****  No Global Variables | | | |
| | 14 | Arithmetic operators | | | |
| | 15 | Comments 20%+ | | 5 | Model as pseudo code |
| | 16 | Named Constants | | | All Local, only Conversions/Physics/Math in Global area |
| | 17 | Programming Style ***** Emulate | | | Emulate style in book/in class repositiory |
| | | | | | |
| 3 | 1 | cin | | | |
| | 2 | Math Expression | | | |
| | 3 | Mixing data types **** | | | |
| | 4 | Overflow/Underflow **** | | | |
| | 5 | Type Casting | | 4 | |
| | 6 | Multiple assignment ***** | | | |
| | 7 | Formatting output | | 4 | |
| | 8 | Strings | | 3 | |
| | 9 | Math Library | | 4 | All libraries included have to be used |
| | 10 | Hand tracing  ****** | | | |
| | | | | | |
| 4 | 1 | Relational Operators | | | |
| | 2 | if | | 4 | Independent if |
| | 4 | If-else | | 4 | |
| | 5 | Nesting | | 4 | |
| | 6 | If-else-if | | 4 | |
| | 7 | Flags ***** | | | |
| | 8 | Logical operators | | 4 | |
| | 11 | Validating user input | | 4 | |
| | 13 | Conditional Operator | | 4 | |
| | 14 | Switch | | 4 | |
| | | | | | |
| 5 | 1 | Increment/Decrement | | 4 | |
| | 2 | While | | 4 | |
| | 5 | Do-while | | 4 | |
| | 6 | For loop | | 4 | |
| | 11 | Files input/output both | | 8 | |
| | 12 | No breaks in loops ****** | | | Failed Project if included |
| | | | | | |
| | | | | | |
| ****** Not required to show | | | Total | 100 | |

# The Code

```
#include <cstdlib>
#include <iostream>
#include <iomanip>
#include <ctime>
#include <cstring>

using namespace std;

/*
 *
 */


int main(int argc, char** argv) {

    // Random number seed
    srand(time(NULL));
    const int HAND_SIZE = 10;
    //const int DECK_SIZE = 108;
    // For loop iterator
    int i;
    int turn = 1; // 1: Players turn | -1: CPU's turn
    //int drwCnt = 0; // Count the number of cards drawn
    int crdPos; // Chosen card position in hand
    char crntCrd[1]; // Current player card
    char crntClr[1]; // Current player card color
    char topCrd[1]; // Card on top of pile
    char topClr[1]; // Card color on top of pile
    char randCrd[1]; // Randomly generated card
    char randClr[1]; // Randomly generated color
    char newCrd[1]; // Top card drawn from deck
```

```
    char newClr[1]; // Top card color drawn from deck
    bool vldCrd = false; // Check if card is in player's hand
    bool endGame = false; // Check for end of game
    string input; // Player input for card
    string colors = "RGBY"; // Available colors to choose from
    // Player and CPU cards as characters
    string p1Crds; // Card
    string p1Clrs; // Color
    string cpuCrds; // Card
    string cpuClrs; // Color
    // Deck cards
    string deck;
    string dckClrs;
    string tmpDeck;
    string tmpClrs;
    // Introduce game
    cout << "Welcome to Uno!\n";
    cout << "You will start with 10 cards!\n\n";
    cout << "Card key:\n";
    cout << "\t0-9 = Card number\n";
    cout << "\tR = Red\n\tG = Green\n\tB = Blue\n\tY = Yellow\n";
    cout << "\tx = Skip next player's turn\n\t+ = Next player draws two
cards\n";
    // Generate standard deck
    for (i = 0; i < 4; i++){
        for (int n = 0; n < 10; n++){
            tmpDeck.append(string (1, char(n + '0')));
            tmpDeck.append(string (1, char(n + '0')));
            tmpClrs.append(string (1, colors[i]));
            tmpClrs.append(string (1, colors[i]));
        }
        for (int n = 0; n < 2; n++){
            tmpDeck.append(string (1, 'x'));
            tmpDeck.append(string (1, '+'));
            tmpClrs.append(string (1, colors[i]));
            tmpClrs.append(string (1, colors[i]));
        }
    }
```

```
    // Shuffle deck
    do {
        int ranPos = rand()%tmpDeck.length();
        deck.append(string (1, tmpDeck[ranPos]));
        dckClrs.append(string (1, tmpClrs[ranPos]));
        tmpDeck.erase(ranPos, 1);
        tmpClrs.erase(ranPos, 1);

    } while (tmpDeck.length() > 0);

    // Give cards to to player and CPU from top of deck
    for (i = 0; i < HAND_SIZE*2; i++){
        newCrd[0] = deck.back();
        newClr[0] = dckClrs.back();
        if (i % 2 == 0){
            p1Crds.append(string (1, newCrd[0]));
            p1Clrs.append(string (1, newClr[0]));
        }
        else if (i % 2 == 1){
            cpuCrds.append(string (1, newCrd[0]));
            cpuClrs.append(string (1, newClr[0]));
        }
        deck.erase(deck.end()-1);
        dckClrs.erase(dckClrs.end()-1);
    }

    // Draw a card from deck to start with
    topCrd[0] = deck.back();
    topClr[0] = dckClrs.back();
    deck.erase(deck.end()-1);
    dckClrs.erase(dckClrs.end()-1);
    // Start game, cycle through turns
    while (!endGame) {
        if (turn == 1){ // Player turn
            do {
                // Process for card selection
                cout << "\nTop card on pile is: \n\t\t[" << topClr[0] <<
topCrd[0] << "]\n";
```

```
cout << "Pick a card from your hand -\n\t\t";
for (i = 0; i < p1Crds.length(); i++){
   cout << setw(4) << i;
}
cout << "\n\t\t";
for (i = 0; i < p1Crds.length(); i++){
   cout << "[" << p1Clrs[i] << p1Crds[i] << "]";
}
cout << " Card: ";
cin >> input;
// Process commands
while (!isdigit(input[0])){
   switch(input[0]){
      case 'd':
         newCrd[0] = deck.back();
         newClr[0] = dckClrs.back();
         p1Crds.append(string (1, newCrd[0]));
         p1Clrs.append(string (1, newClr[0]));
         deck.erase(deck.end()-1);
         dckClrs.erase(dckClrs.end()-1);
         break;
   }
   cout << "Updated hand:\t";
   for (i = 0; i < p1Crds.length(); i++)
      cout << setw(4) << i;
   cout << "\n\t\t";
   for (i = 0; i < p1Crds.length(); i++)
      cout << "[" << p1Clrs[i] << p1Crds[i] << "]";
   cout << " Card: ";
   cin >> input;
}
// End processing commands
crdPos = stoi(input);
crntCrd[0] = p1Crds[crdPos];
crntClr[0] = p1Clrs[crdPos];

// Process action cards
switch(crntCrd[0]){
```

```
        // Draw 2: Make CPU draw two cards
        case '+':
            for (i = 0; i < 2; i++){
                newCrd[0] = deck.back();
                newClr[0] = dckClrs.back();
                cpuCrds.append(string (1, newCrd[0]));
                cpuClrs.append(string (1, newClr[0]));
                deck.erase(deck.end()-1);
                dckClrs.erase(dckClrs.end()-1);
            }
            cout << "CPU grabbed two cards from deck!\n";
            break;
        // Skip turn: Skip CPU's turn
        case 'x':
            turn *= -1;
            cout << "CPU turn skipped!\n";
            break;
    }
    // End card selection

    // Check selected card against top card
    if (crntCrd[0] == topCrd[0] || crntClr[0] == topClr[0]){
        vldCrd = true;
    }
} while (!vldCrd);

// Place card on top of pile
topCrd[0] = crntCrd[0];
topClr[0] = crntClr[0];

// Remove placed card from hand
p1Crds.erase(crdPos, 1);
p1Clrs.erase(crdPos, 1);

// End turn
if (p1Crds.length() == 0)
    endGame = true;
turn *= -1;
```

```
    vldCrd = false;

} else if (turn == -1) { // CPU turn
    cout << "CPU's turn\n";
    do {
        // Find card in hand that matches
        for (i = 0; i < cpuCrds.length(); i++){
            if (!vldCrd){
                if (cpuClrs[i] == topClr[0] || cpuCrds[i] == topCrd[0]){
                    crdPos = i;
                    vldCrd = true;
                }
            }
        }

        // If no card matches, draw from deck
        if (!vldCrd){
            newCrd[0] = deck.back();
            newClr[0] = dckClrs.back();
            cpuCrds.append(string (1, newCrd[0]));
            cpuClrs.append(string (1, newClr[0]));
            deck.erase(deck.end()-1);
            dckClrs.erase(dckClrs.end()-1);
            cout << "CPU grabbed card from deck\n";
        }
    } while (!vldCrd);

    // Process action cards
    switch(cpuCrds[crdPos]){
        // Draw 2: Make Player draw two cards
        case '+':
            for (i = 0; i < 2; i++){
                newCrd[0] = deck.back();
                newClr[0] = dckClrs.back();
                p1Crds.append(string (1, newCrd[0]));
                p1Clrs.append(string (1, newClr[0]));
                deck.erase(deck.end()-1);
                dckClrs.erase(dckClrs.end()-1);
```

```
                }
                cout << "Player grabbed two cards from deck!\n";
                break;
            // Skip turn: Skip Player's turn
            case 'x':
                turn *= -1;
                cout << "Player turn skipped!\n";
                break;
        }

        // Place card on top of pile
        topCrd[0] = cpuCrds[crdPos];
        topClr[0] = cpuClrs[crdPos];

        // Remove placed card from hand
        cpuCrds.erase(crdPos, 1);
        cpuClrs.erase(crdPos, 1);

        // End turn
        cout << "CPU has " << cpuCrds.length() << " cards left." <<
endl;
        if (cpuCrds.length() == 0)
            endGame = true;
        turn *= -1;
        vldCrd = false;

    }
  }

  if (p1Crds.length() == 0)
    cout << endl << "You won!!" << endl;
  else
    cout << endl << "CPU won!!" << endl;

  return 0;
}
```