PROGRAMMING FUNDAMENTALS

CT-175

PROJECT NAME: UNO CARDS

FOR

First Year

Computer Sciences

BATCH: 2023-24

GROUP MEMBERS:

1. **MISL-E-NOOR OVEIS, CT-073**
2. **MUHAMMAD ALI, CT-074**
3. **SYED MUHAMMAD ABDUL REHMAN, CT-092**
4. **EISHA SOHAIL, CT-068**

DISCIPLINE: FSCS (COMPUTER SCIENCE)

SECTION: B

SEMESTER: FALL 2023-24

DEPARTMENT OF CSIT

NED UNIVERSITY OF ENGINEERING & TECHNOLOGY, KARACHI, PAKISTAN.

TABLE OF CONTENTS:

1. INTRODUCTION:

The Uno Card Game project aims to provide a console-based implementation of the classic Uno card game for two players. By recreating the Uno experience in a digital format, our objective is to offer an interactive and engaging platform for players to enjoy this popular card game.

1.1 PROBLEM STATEMENT:

In the absence of a console-based Uno card game for interactive two-player gaming, there is a need to fill this gap and provide a virtual environment for players to enjoy Uno. This project addresses the lack of a digital Uno game by implementing a console version that captures the essence of the game.

1.2 PROJECT DESCRIPTION:

The Uno Card Game project is a console-based application developed in C. It simulates the Uno card game, featuring numeric cards in four colors, special action cards (Reverse, Skip, Draw Two), and wildcard cards (Wild, Wild Draw Four). The purpose is to create an enjoyable and functional representation of Uno within a console.

2. CONCEPTS COVERED:

- **ARRAYS:** Arrays are used to represent the deck of Uno cards, player hands, and various card attributes.
- **FUNCTIONS:** The project demonstrates the use of functions to encapsulate different aspects of the game, promoting modularity and code organization.
- **CONTROL STRUCTURES**: Control structures like loops and conditionals are extensively used to implement the game flow, turn handling, and card validation.
- **FILE I/O (STANDARD INPUT/OUTPUT):** Standard Input/Output functions are employed for user interaction in the console environment.

3. FEATURES:

3.1 INITIALIZATION:

The program initializes the deck with Uno cards, assigns cards to players, and sets up the initial game state.

3.2 PLAYERS TURN:

Players take turns, and each turn involves playing a card based on matching colour or number rules

3.3 CARD VALIDATION:

The program validates whether the thrown card is a valid move, considering colour, number, and special action rules.

3.4 WILD CARD HANDLING:

Special attention is given to Wild and Wild Draw Four cards, allowing players to choose a new colour.

3.5 PENALTIES:

Penalties are implemented, such as skipping a turn or drawing additional cards, based on specific actions or rules.

3.6 END OF GAME:

The game continues until one player successfully empties their hand, declaring them the winner.

4. IMPLEMENTATION DETAILS:

The code is structured into functions to enhance modularity and readability. Noteworthy sections include:

4.1 player1Turn():

Handles Player 1's turn, validating card plays, and applying card effects.

4.2 player2Turn():

Manages Player 2's turn with similar functionality.

4.3 processCards():

Governs the main game loop and turn-switching logic.

5. SOURCE CODE:

```
#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <string.h>


// ANSI color escape codes

#define ANSI_COLOR_YELLOW "\x1b[33m"

#define ANSI_COLOR_RED "\x1b[31m"

#define ANSI_COLOR_RESET "\x1b[0m"


// prototypes

void playerCards();

void randomCard();

void processCards();

void player1Turn();

void player2Turn();


// main deck

char cards[60][5] = {

    "R0", "R1", "R2", "R3", "R4", "R5", "R6", "R7", "R8", "R9", "G0",
"G1", "G2", "G3", "G4", "G5", "G6", "G7", "G8", "G9",

    "Y0", "Y1", "Y2", "Y3", "Y4", "Y5", "Y6", "Y7", "Y8", "Y9",
"B0", "B1", "B2", "B3", "B4", "B5", "B6", "B7", "B8", "B9",
    "R+2", "RR", "RS", "W", "D4", "G+2", "GR", "GS", "W", "D4",
"Y+2", "YR", "YS", "W", "D4", "B+2", "BR", "BS", "W", "D4"};


// all variables
int size1 = 7, size2 = 7;
char player1Cards[7][5], player2Cards[7][5];
char currentCard[5];
char throwCard[5];
char color[0];
int skip = 0;
int unoCalledPlayer1 = 0, unoCalledPlayer2 = 0;


int main()
{
    printf("\n ---------------------------- CAPSLOCK SHOULD BE
ENABLE -------------------\nRules:\n1. Each player will be distributed
7 cards\n\n2. Player 1 will get the first turn.\n\n3. Each player has to
throw the card according to either the color or the number of the last
card. If you don't have the specific card, you have to type
(SKIP)\n\n4. After you enter SKIP, a new card will be drawn and
automatically added to your deck\n\n5. When only 1 card will be left
for either of the players, you will have to enter UNO just before
playing the last card\n\n6. If you don't enter UNO before entering
your last card, it won't be allowed to play, and a new card will be
added to your deck as a punishment\n\n\n\n---------------- Have Fun ---
-------------\n\n\n\n");
    srand(time(NULL));
    randomCard();
    playerCards();


    printf("First card from the deck:\n");
    printf("-----------------\t%s\t-------------------------\n", currentCard);
    for (int i = 0; i < size1; i++)
    {
        printf("\nCard %d Of Player 1 : %s%s%s", i + 1,
ANSI_COLOR_YELLOW, player1Cards[i],
ANSI_COLOR_RESET);
    }


    printf("\n----------------------------------------");
    for (int i = 0; i < size2; i++)
    {
        printf("\nCard %d Of Player 2: %s%s%s", i + 1,
ANSI_COLOR_RED, player2Cards[i], ANSI_COLOR_RESET);
    }


    processCards();
    return 0;
}

void playerCards()
{
    // cards to player1
    for (int i = 0; i < size1; i++)
    {
        int random;
        do
        {
            random = rand() % 60;
        } while (strcmp(cards[random], " ") == 0);
        strcpy(player1Cards[i], cards[random]);
        strcpy(cards[random], " ");
    }
    // cards to player2
    for (int i = 0; i < size2; i++)
    {
        int random;
        do
        {
            random = rand() % 60;
        } while (strcmp(cards[random], " ") == 0);

        strcpy(player2Cards[i], cards[random]);
        strcpy(cards[random], " ");
    }
}
void randomCard()
{
    int random;
    do
    {
        random = rand() % 40;
    } while (strcmp(cards[random], " ") == 0);
    strcpy(currentCard, cards[random]);
    strcpy(cards[random], " ");
}

void player1Turn()
{
    int found = 0;
    int validthrow = 0;

     // Check if UNO needs to be called
    if (size1 == 1 && unoCalledPlayer1 == 0) {

        char unoInput[4];
        scanf("%3s", unoInput);
        if (strcmp(unoInput, "UNO") == 0) {
            unoCalledPlayer1 = 1;
        } else {
            printf("You must say UNO before playing your last card.
Penalty: Player 2 wins!\n");
            printf("Player 1 failed to say UNO and loses the game.\n");
            // End the game, Player 2 wins
            exit(0);
        }
    }

    if (skip == 1)
    {
        skip = 0;
        return;
    }

    while (validthrow == 0)
    {
        printf("\nPlayer 1: ");
        scanf("%4s", throwCard);
//skip check
        if (strcmp(throwCard, "SKIP") == 0)
        {
            skip = 1;
            printf("\n----------------------------\n");
            printf("Player 1 has skipped his turn.\n");
            printf("\n----------------------------\n");
            printf("\n1 Card is added to player 1's deck\n");
            for (int i = 0; i < 1; i++)
            {
                int random;
                do
                {
                    random = rand() % 60;
                } while (strcmp(cards[random], " ") == 0);

                strcpy(player1Cards[size1], cards[random]);

                strcpy(cards[random], " ");
            }
            size1++;
            for (int i = 0; i < size1; i++)
            {
                printf("Card %d of player 1 : %s%s%s\n", i + 1,
ANSI_COLOR_YELLOW, player1Cards[i],
ANSI_COLOR_RESET);
            }
            return player2Turn();
        }

        else
        {
            for (int i = 0; i < size1; i++)
            {
                // Checking that card is in the deck or not----------
                if (strcmp(player1Cards[i], throwCard) == 0)
```

```
                {
                    found = 1;
                    // Shift the remaining cards to the left starting from the
index i
                    for (int j = i; j < size1 - 1; j++)
                    {
                        strcpy(player1Cards[j], player1Cards[j + 1]);
                    }
                    if (throwCard[0] == currentCard[0] || throwCard[1] ==
currentCard[1] || throwCard[1] == '+' || strcmp(throwCard, "D4") == 0
|| strcmp(throwCard, "W") == 0 || throwCard[1] == 'R' || throwCard[1]
== 'S' )
                    {
                        size1--;
                        // Decrement size to reflect the removal of the card
                    }
                    break;
                }
            }
 // main terminal--------------------
            if (found == 1 && (throwCard[0] == currentCard[0] ||
throwCard[1] == currentCard[1] || throwCard[1] == '+' ||
strcmp(throwCard, "D4") == 0 || throwCard[1] == 'R' || throwCard[1]
== 'S' || strcmp(throwCard, "W") == 0))
            {
 // REVERSE CHECK----------------------------------
```

```
        if (throwCard[0] == currentCard[0] && (throwCard[1] ==
'R' || throwCard[1] == 'S'))
        {
        printf("Valid card! Player 1 can throw this card.\n");
        printf("Player 1's deck after throwing card:\n");
        for (int i = 0; i < size1; i++)
        {
            printf("Card %d of player 1 : %s%s%s\n", i + 1,
ANSI_COLOR_YELLOW, player1Cards[i],
ANSI_COLOR_RESET);
        }
            printf("\nPlayer one either reversed or skipped the turn.");
            return player1Turn();
        }
// invalid reverse---------------------------------------------
        if (throwCard[0] != currentCard[0] && (throwCard[1] ==
'R' || throwCard[1] == 'S'))
        {
            printf("Invalid card! Player 1 cannot throw this card. The
card should match the color or number.\n Or it should be a special
card.\n");
            printf("Try Again \n");
            return player1Turn();
        }

 // checking for Draw2-----------------------------------------
```

```
        if (throwCard[0] == currentCard[0] && throwCard[1] ==
'+')
        {
          printf("\n---------------------------\n");
          printf("Player 2 will draw 2 cards");
          printf("\n---------------------------\n");
          for (int i = 0; i < 2; i++)
          {
            int random;
            do
            {
              random = rand() % 60;
            } while (strcmp(cards[random], " ") == 0);

            strcpy(player2Cards[size2], cards[random]);
            size2++;
            strcpy(cards[random], " ");
          }
// showing player 2's deck with additiona; cards--------------------------
---
          printf("\n---------------------------\n");
          printf("Player 2's deck after drawing 2 cards:\n");
          printf("\n---------------------------\n");

          for (int i = 0; i < size2; i++)
```

```
                {
                    printf("Card %d of player 2 : %s%s%s\n", i + 1,
ANSI_COLOR_RED, player2Cards[i], ANSI_COLOR_RESET);
                }
                currentCard[0] = throwCard[0];


                printf("-----------------\t %s \t---------------\n", throwCard);


            }
//invalid draw 2-----------------------------------------------
            else if(throwCard[0] != currentCard[0] && throwCard[1]
== '+')
            {
                printf("Invalid card! Player 1 cannot throw this card. The
card should match the color or number.\n Or it should be a special
card.\n");
                printf("Try Again \n");
                return player1Turn();
            }
 // CHECKING FOR D4-------------------------------------------------------
--------------
            if (strcmp(throwCard, "D4") == 0)
            {
                printf("\n----------------------------\n");
                printf("Player 2 will draw 4 cards");
                printf("\n----------------------------\n");
```

```
        for (int i = 0; i < 4; i++)
        {
            int random;
            do
            {
                random = rand() % 60;
            } while (strcmp(cards[random], " ") == 0);

            strcpy(player2Cards[size2], cards[random]);
            size2++;
            strcpy(cards[random], " ");
        }

        printf("\n---------------------------\n");
        printf("Player 2's deck after drawing 4 cards:\n");
        printf("\n---------------------------\n");

        for (int i = 0; i < size2; i++)
        {
            printf("Card %d of player 2 : %s%s%s\n", i + 1,
ANSI_COLOR_RED, player2Cards[i], ANSI_COLOR_RESET);
        }
// ask for color input
        printf("Choose a color (R, G, B, Y): ");
        while (1)
```

```
                    {
                        fflush(stdin);
                        scanf(" %c", &color[0]);
                        if (color[0] == 'R' || color[0] == 'G' || color[0] == 'B' ||
color[0] == 'Y')
                        {
                            printf("Chosen color:%c", color[0]);
                            currentCard[0] = color[0];
                            break;
                        }
                        else
                        {
                            printf("Invalid color choice. Please choose R, G, B,
or Y: ");
                        }
                    }

                    printf("\n-----------------\t %c \t---------------\n", color[0]);

                }
// check for wild card-------------------------------------
                else if (strcmp(throwCard, "W") == 0) {
                printf("Choose a color (R, G, B, Y): ");
                while (1) {
                    fflush(stdin);
                    scanf(" %c", &color[0]);
                    if (color[0] == 'R' || color[0] == 'G' || color[0] == 'B' ||
color[0] == 'Y') {
                        printf("\nChosen Color : %c", color[0]);
                        currentCard[0] = color[0];
                        break;
                    } else
                    {
                        printf("Invalid color choice. Please choose R, G, B, or
Y: ");
                    }
                }
                }
                else
                {
                    // Update the current card on the table
                    strcpy(currentCard, throwCard);
                    printf("\n-----------------\t %s \t---------------\n",
currentCard);
                }

                printf("\nValid card! Player 1 can throw this card.\n");
                printf("Player 1's deck after throwing card:\n");
                for (int i = 0; i < size1; i++)
                {
                    printf("Card %d of player 1 : %s%s%s\n", i + 1,
ANSI_COLOR_YELLOW, player1Cards[i],
ANSI_COLOR_RESET);
                }

                validthrow = 1;
            }
            else
            {
                printf("Invalid card! Player 1 cannot throw this card. The
card should match the color or number.\n Or it should be a special
card.\n");
                printf("Try Again \n");
            }
        }
      }
}

void player2Turn()
{
      int found = 0;
      int validthrow1 = 0;
       // Check if UNO needs to be called
      if (size2 == 1 && unoCalledPlayer2 == 0) {
        fflush(stdin);
        char unoInput[4];
        scanf("%3s", unoInput);
        if (strcmp(unoInput, "UNO") == 0) {
            unoCalledPlayer2 = 1;
        } else {
            printf("You must say UNO before playing your last card.
\nPenalty: Player 1 wins!\n");
            printf("Player 2 failed to say UNO and loses the game.\n");
            // End the game, Player 1 wins
            exit(0);
        }
    }
    if (skip == 1)
    {
        skip = 0;
        return;
    }

    while (validthrow1 == 0)
    {
        printf("\nPlayer 2: ");
        scanf("%4s", throwCard);
//skip check-----------------------
        if (strcmp(throwCard, "SKIP") == 0)
```

```
{
    skip = 1;
    printf("\n---------------------------\n");
    printf("Player 2 has skipped his turn.\n");
    printf("\n---------------------------\n");
    printf("\n1 Card is added to player 2's deck\n");
    for (int i = 0; i < 1; i++)
    {
        int random;
        do
        {
            random = rand() % 60;
        } while (strcmp(cards[random], " ") == 0);

        strcpy(player2Cards[size2], cards[random]);
        size2++;
        strcpy(cards[random], " ");
    }
    for (int i = 0; i < size2; i++)
    {
        printf("Card %d of player 2 : %s%s%s\n", i + 1,
ANSI_COLOR_RED, player2Cards[i], ANSI_COLOR_RESET);
    }
    return player1Turn();
```

```
        }
        else
        {
        for (int i = 0; i < size2; i++)
        {
            if (strcmp(player2Cards[i], throwCard) == 0)
            {
                found = 1;
                // Shift the remaining cards to the left starting from the
index i
                for (int j = i; j < size2 - 1; j++)
                {
                    strcpy(player2Cards[j], player2Cards[j + 1]);
                }
                if (throwCard[0] == currentCard[0] || throwCard[1] ==
currentCard[1] || throwCard[1] == '+' || strcmp(throwCard, "D4") ==
0||strcmp(throwCard, "W") == 0 || throwCard[1] == 'R' ||
throwCard[1] == 'S' )
                {
                    size2--; // Decrement size to reflect the removal of the
card
                }
                break;
            }
        }
// main terminal--------------------
```

```
        if (found == 1 && (throwCard[0] == currentCard[0] ||
throwCard[1] == currentCard[1] || throwCard[1] == '+' ||
strcmp(throwCard, "D4") == 0 || throwCard[1] == 'R' || throwCard[1]
== 'S' || strcmp(throwCard, "W") == 0))

        {

// REVERSE CHECK---------------------------------

            if (throwCard[0] == currentCard[0] && (throwCard[1] == 'R'
|| throwCard[1] == 'S'))

            {

                 printf("Valid card! Player 1 can throw this card.\n");

                 printf("Player 1's deck after throwing card:\n");

                for (int i = 0; i < size2; i++)

                {

                    printf("Card %d of player 2 : %s%s%s\n", i + 1,
ANSI_COLOR_RED, player2Cards[i], ANSI_COLOR_RESET);

                }

                    printf("\nPlayer two either reversed or skipped the
turn.");

                    return player2Turn();

            }

// invalid reverse--------------------------------------------

            if (throwCard[0] != currentCard[0] && (throwCard[1] == 'R' ||
throwCard[1] == 'S'))

            {

                printf("Invalid card! Player 2 cannot throw this card. The
card should match the color or number.\n Or it should be a special
card.\n");
```

```
            printf("Try Again \n");
            return player2Turn();
        }

// checking for Draw2----------------------------------
        if (throwCard[0] == currentCard[0] && throwCard[1] == '+')
        {

            printf("\n---------------------------\n");
            printf("Player1 will draw 2 cards");
            printf("\n---------------------------\n");
            for (int i = 0; i < 2; i++)
            {
                int random;
                do
                {
                    random = rand() % 60;
                } while (strcmp(cards[random], " ") == 0);

                strcpy(player1Cards[size1], cards[random]);
                size1++;
                strcpy(cards[random], " ");
            }
// showing player 2's deck with additiona; cards--------------------------
---
```

```
                printf("\n----------------------------\n");
                printf("Player 1's deck after drawing 2 cards:\n");
                printf("\n----------------------------\n");

                for (int i = 0; i < size1; i++)
                {
                    printf("Card %d of player 1 : %s%s%s\n", i + 1,
ANSI_COLOR_YELLOW, player1Cards[i],
ANSI_COLOR_RESET);
                }
                currentCard[0] = throwCard[0];

                printf("-----------------\t %s \t---------------\n", throwCard);
            }
 //invalid draw 2-----------------------------------------------
             else if(throwCard[0] != currentCard[0] && throwCard[1] ==
'+')
            {
                printf("Invalid card! Player 2 cannot throw this card. The
card should match the color or number.\n Or it should be a special
card.\n");
                    printf("Try Again \n");
                    return player2Turn();
            }
// CHECKING FOR D4-----------------------------
            if (strcmp(throwCard, "D4") == 0)
```

```
{
    printf("\n----------------------------\n");
    printf("Player1 will draw 4 cards");
    printf("\n----------------------------\n");
    for (int i = 0; i < 4; i++)
    {
        int random;
        do
        {
            random = rand() % 60;
        } while (strcmp(cards[random], " ") == 0);

        strcpy(player1Cards[size1], cards[random]);
        size1++;
        strcpy(cards[random], " ");
    }

    printf("--------------------------------\n");
    printf("Player 1's deck after drawing 4 cards:\n");
    printf("--------------------------------\n");

    for (int i = 0; i < size1; i++)
    {
```

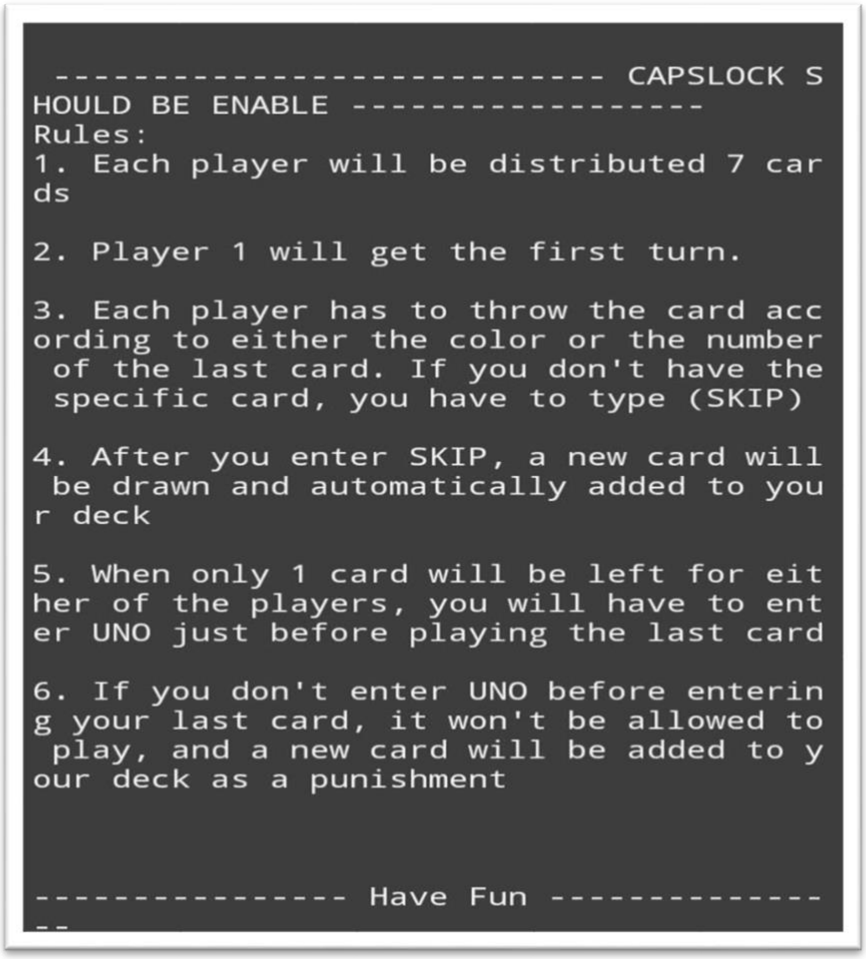
```
                printf("Card %d of player 1 : %s%s%s\n", i + 1,
ANSI_COLOR_YELLOW, player1Cards[i],
ANSI_COLOR_RESET);
            }
//ask for input COLOR------------------------
            printf("Choose a color (R, G, B, Y): ");
            while (1)
            {
                fflush(stdin);
                scanf(" %c", &color[0]);
                if (color[0] == 'R' || color[0] == 'G' || color[0] == 'B' ||
color[0] == 'Y')
                {
                    printf("Chosen color: %c", color[0]);
                    currentCard[0] = color[0];
                    break;
                }
                else
                {
                    printf("Invalid color choice. Please choose R, G, B, or
Y: ");
                }
            }

            printf("\n-----------------\t %c \t--------------\n", color[0]);
        }
```

```
// check for wild card
        else if (strcmp(throwCard, "W") == 0) {
            printf("Choose a color (R, G, B, Y): ");
            while (1)
            {
                fflush(stdin);
                scanf(" %c", &color[0]);
                if (color[0] == 'R' || color[0] == 'G' || color[0] == 'B' ||
color[0] == 'Y') {
                    printf("Chosen Color : %c", color[0]);
                    currentCard[0] = color[0];
                    break;
                } else {
                    printf("Invalid color choice. Please choose R, G, B, or
Y: ");
                }
            }
        }
        else
        {
            // Update the current card on the table
            strcpy(currentCard, throwCard);
            printf("-----------------\t %s \t---------------\n", currentCard);
        }
```

```
            printf("Valid card! Player 2 can throw this card.\n");
            printf("Player 2's deck after throwing card:\n");
            for (int i = 0; i < size2; i++)
            {
                printf("Card %d of player 2 : %s%s%s\n", i + 1,
ANSI_COLOR_RED, player2Cards[i], ANSI_COLOR_RESET);
            }

            validthrow1 = 1;
        }
        else
        {
            printf("Invalid card! Player 2 cannot throw this card. The card
should match the color or number.\n");
            printf("Try Again\n");
        }
    }}
}

void processCards()
{
    int currentplayer = 1;
    while (1)
    {
        if (currentplayer == 1)
        {
            player1Turn();
            if (size1 == 0)
            {
                printf("Player 1 Wins");
                break;
            }
        }
        else
        {
            player2Turn();
            if (size2 == 0)
            {
                printf("Player 2 Wins");
                break;
            }
        }
        if (currentplayer == 1) {
            currentplayer = 2;
        } else {
            currentplayer = 1;
        }

    }
}
```

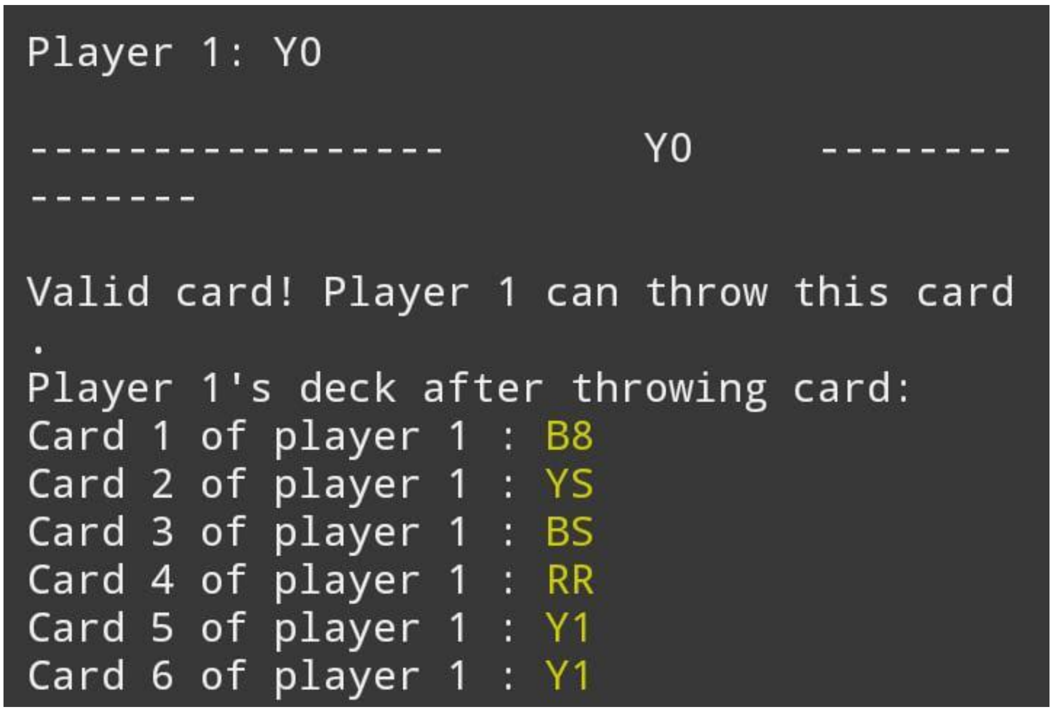
6. OUTPUT:

GAME RULES:



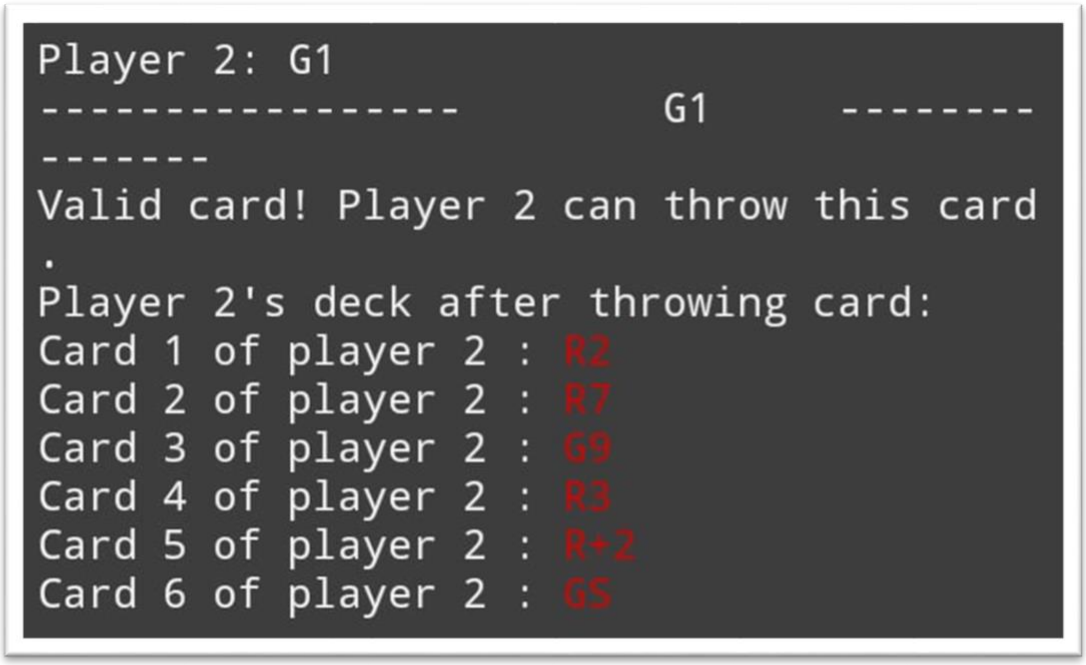
```
 --------------------------- CAPSLOCK S
HOULD BE ENABLE ------------------
Rules:
1. Each player will be distributed 7 car
ds

2. Player 1 will get the first turn.

3. Each player has to throw the card acc
ording to either the color or the number
 of the last card. If you don't have the
 specific card, you have to type (SKIP)

4. After you enter SKIP, a new card will
 be drawn and automatically added to you
r deck

5. When only 1 card will be left for eit
her of the players, you will have to ent
er UNO just before playing the last card

6. If you don't enter UNO before enterin
g your last card, it won't be allowed to
 play, and a new card will be added to y
our deck as a punishment



--------------- Have Fun -------------
--
```

<u>PLAYER 1:</u>



```
Player 1: Y0

-----------------        Y0     --------
-------

Valid card! Player 1 can throw this card
.
Player 1's deck after throwing card:
Card 1 of player 1 : B8
Card 2 of player 1 : YS
Card 3 of player 1 : BS
Card 4 of player 1 : RR
Card 5 of player 1 : Y1
Card 6 of player 1 : Y1
```

PLAYER 2:



```
Player 2: G1
-----------------          G1       --------
-------
Valid card! Player 2 can throw this card
.
Player 2's deck after throwing card:
Card 1 of player 2 : R2
Card 2 of player 2 : R7
Card 3 of player 2 : G9
Card 4 of player 2 : R3
Card 5 of player 2 : R+2
Card 6 of player 2 : GS
```

7. GROUP MEMBER CONTRIBIUTIONS

7.1 Mislenoor Oveis:

Responsibility: Deck Initialization and Card

Dealing Contribution:

- Implemented the initialization of the Uno deck with the appropriate cards.
- Developed the logic for dealing seven cards to each player.
- Ensured the proper setup of the game environment for a smooth start.

7.2 Muhammad Ali:

Responsibility: Player Turn Handling and Card

Validation Contribution:

- Implemented the functionality for handling Player Turns, ensuring a sequential gameplay flow.
- Developed card validation mechanisms to check if a played card is valid based on color, number, or special action.
- Ensured that the game adheres to Uno rules during player interactions.

7.3 Syed Muhammad Abdul Rehman:

Responsibility: Special Action Card Functionality and Game

Flow Contribution:

- Implemented the unique effects of special action cards (Reverse, Skip, Draw Two).
- Developed the game flow and turn-switching logic to maintain a cohesive gaming experience.
- Ensured that the game progresses smoothly, considering special actions and player turns.

7.4 Eisha Sohail:

Responsibility: Code Review, Refactoring, and Documentation

Contribution:

- Conducted thorough code reviews to identify and address potential issues or improvements.
- Participated in refactoring sessions to enhance code readability, modularity, and adherence to best practices.
- Collaborated with other team members to optimize code efficiency.
- Compiled comprehensive documentation for the Uno Card Game codebase.
- Drafted the project report, ensuring clarity, organization, and inclusion of all essential information.
- Collaborated with other team members to gather and integrate individual contributions.

8. **LIMITATIONS/FUTURE ENHANCEMENTS**:

8.1 LIMITATIONS:

- Limited extensibility for accommodating changes or extensions.
- Console-based interface limits graphical elements and user experience.
- Minimal error handling for unexpected inputs.

8.2 FUTURE ENHANCEMENTS:

- Implement a graphical user interface for a more user-friendly experience.
- Add opponents for single-player mode.
- Enhance error handling for improved reliability.
- Explore scalability for accommodating more players.

9. **CONCLUSION:**

In conclusion, the Uno Card Game project successfully achieved its objectives by providing a functional and enjoyable representation of the classic Uno card game within a console environment. The implementation demonstrated proficiency in procedural programming techniques while acknowledging areas for potential improvements and future enhancements.

THANKYOU!