Title

UNO! V.6
Card Game

Course

CIS-17A

Section

43396

Due Date

May 17, 2021

Author

Lindsay Kislingbury

1 Introduction

Title: UNO!

Uno is a shedding-type card game. The player attempts to get rid of cards in their hand while adding cards to other players' hands. A player wins when there are no cards left in their hand. There are several "action" cards in UNO which change the play direction, change the color of discard, and add/remove cards from players' hands. This program utilizes vectors, pointers, and dynamically allocated arrays to continuously change these elements, reducing processing time and number of operations.

1.1 Summary:

Project Size: ~750 lines

This project demonstrates the concepts covered in Chapters 9 through 12 as well as many of the concepts covered in CIS-5.

- Pointers are utilized heavily in this program. Every hand played required that the player's game data be passed by pointer to several functions to perform operations on it.
- The array of Player structures containing all player game data is dynamically allocated after getting the number of players from the user, filled by the deal function, and returned to the main game function.
- String member functions were utilized for input validation and for converting the name of each player to a char array for writing to a binary file.
- Nested Structures and enumerated data types were used to group data, pass-to and return-from functions, and write to binary file.
- Various control flow structures covered in CIS-5 were utilized in this program.

There is much more that can be done with this program. I really wanted to implement an existing player system which would pull data for a previous player and modify the number of games played, wins, and highest scores. This would allow a user to enter their name and pull their past game data from the binary file. After having some difficulty writing and reading player names from the binary file, I ran out of time. I plan on implementing this feature in Version 2.

2. UNO! Rules

Official Rules published by Mattel can be found here: https://service.mattel.com/instruction_sheets/42001pr.pdf

Setup:
The game is for 2-10 players.
Every player starts with 7 cards.
The rest of the cards are placed in the draw pile.
A discard pile is created by flipping over a card from the draw pile.
If the top card is a Wild or Wild Draw 4, it is returned to the deck and another card is flipped.

Cards:
108 Cards

Number Cards	Action Cards:
19 Blue Cards: 0 to 9	8 Skip Cards: 2 each in Blue, Green, Red and Yellow
19 Green Cards: 0 to 9	8 Draw-2 Cards: 2 each in Blue, Green, Red and Yellow
19 Red Cards: 0 to 9	8 Wild Cards
19 Yellow Cards: 0 to 9	8 Wild Draw-4 Cards

Game Play:

- Players examine their card and try to match the top card to the discard.
- Cards are matched by color, number, or action.
  For example, if the discard is blue 5, a player has the option of playing any blue card or any color card with a 5.
- Wild cards may be played at any time and the player may choose to the change the leading color with it.
- If the player does not have a playable card, they must draw from the draw pile.
- If the card drawn can be played, the player must play it.

Note: If the draw pile is exhausted, the draw pile is shuffled and becomes the new draw pile. Play continues on the single card from discard as normal.

- Play continues until a player has a single card.
- The moment a player has just one card they must call “UNO!” If they do not call “UNO!” before the next player has taken their turn, that player must draw two new cards as penalty.
  Calling “UNO!” needs to be repeated every time a player is left with one card.
- Once a player has no cards remaining, the game is over and points are scored.

Action Cards:

Reverse Switch the direction of turns. If the play was moving left, it moves right.

Skip The next player’s turn is skipped.

Draw-2 The next player must draw 2 cards.

Wild This card can be used to represent any color and can be placed on any card.
The player chooses which color it will represent for the next player’s turn.

Wild Draw-4 Acts just like a Wild card except that the next player also has to draw 4 cards.

Scoring:
When a player no longer has any cards the game ends and that player is the winner.
The winner receives points for the cards left in all other players’ hands.

Points:

All Number Cards	Face Value	Reverse	20 Points
Draw-2	20 Points	Wild	50 Points
Skip	20 Points	Wild Draw-4	50 Points

3 Development

3.1 ADTs

Type enumerated data type

```
enum Type{NUMBER,SKIP,REVERSE,DRAW2,WILD,WILD4};
```

Primarily utilized in Switch statements when switching between card types.

Card structure

```
struct Card{
    Type                type;    //Type of Card
    short               num;     //Number
    char                color;   //Color
    string              colName; //Color Name
    int                 points;  //Points
};
```

Holds all related data for a single card.

Player structure

```
struct Player{
    vector<Card>   data;    //Hand of Cards
    string         player;  //Player Name
    int            gmScr,   //Ending Game Score
                   lrgHnd,  //Largest Hand
                   numHnds; //Number of Hands Played
    bool           winner;  //Hold Win Status
    vector<int>    hndSzs;  //Hold the size of every hand played
};
```

Holds all game data for a single player, including a vector of Card structures.

Scores structure

```
struct Scores{
    char                winner[25]; //Name of the game winner
    int                 numHnds;    //Total number of hands played
    int                 lrgHnd;     //Single largest hand played
    int                 score;      //Winner score
    int                 numPlyrs;   //Number of players
};
```

Holds the end game data, which is calculated after a game is complete. The elements of this structure are written to a binary file.

3.2 play()

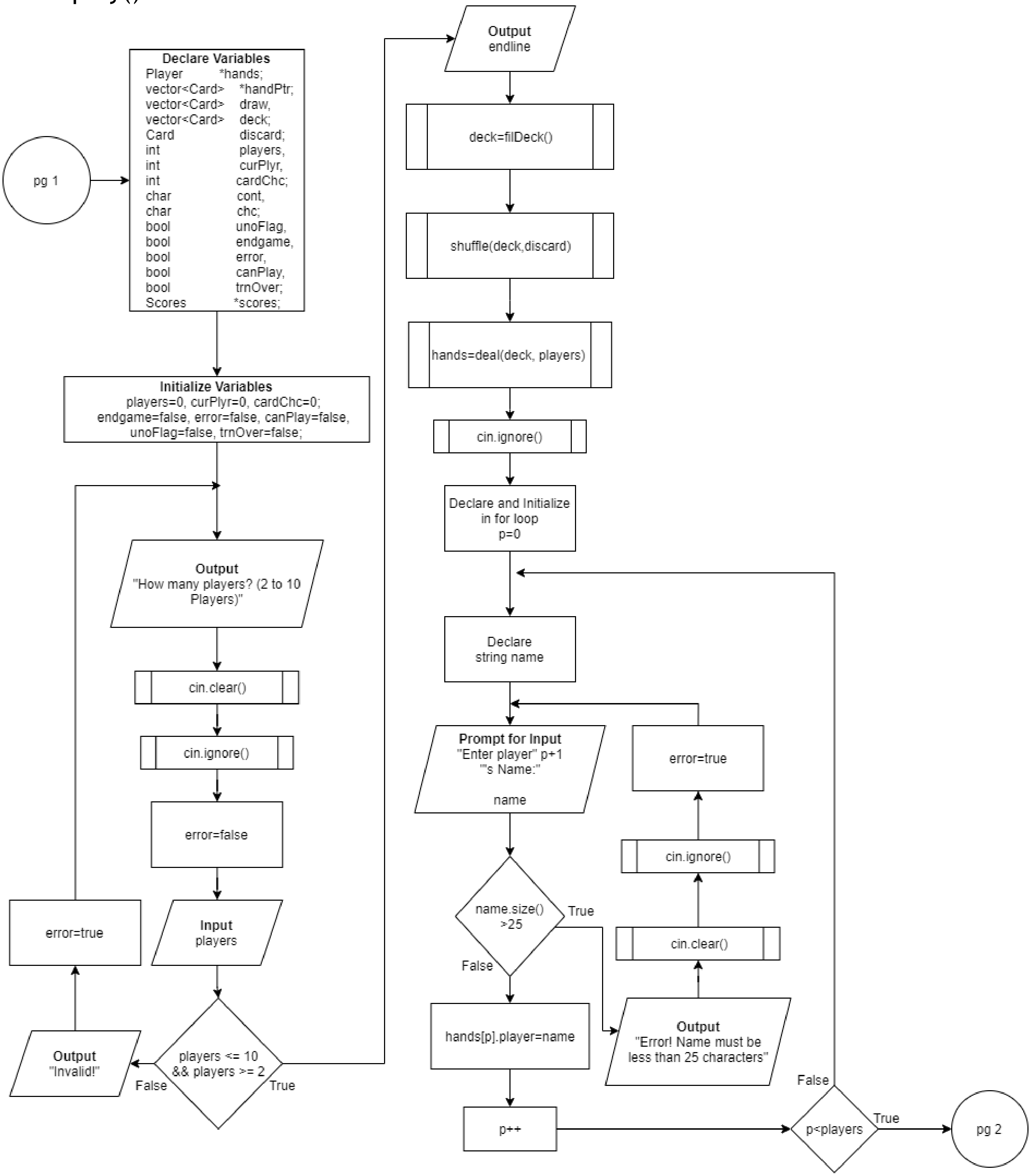
3.2.1 play() Pseudo Code:

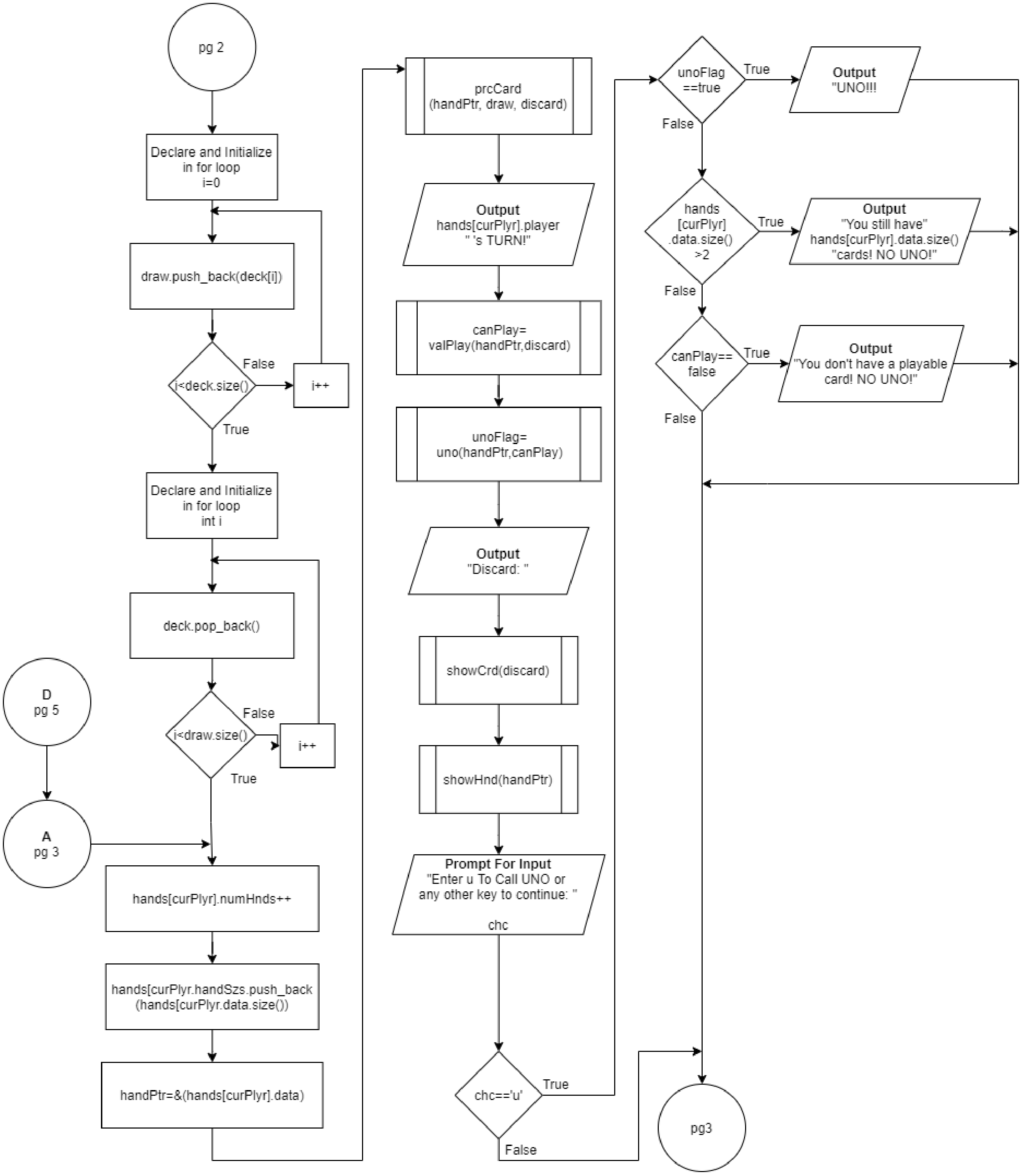
```
void play()
***************************************************************************
VARIABLES:
    Player          *hands;    //Pointer to an array of player hand vectors
    vector<Card>    *handPtr;  //Pointer to current player's hand
    vector<Card>    draw,      //Draw Pile
                    deck;      //The Entire Deck of Cards
    Card            discard;   //Discard Pile
    int             players,   //Number of players
                    curPlyr,   //Current player
                    cardChc;   //Card choice
    char            cont,      //User input to continue after message
                    chc;       //Player choice to call uno or play a card
    bool            unoFlag,   //Flag, if player called uno
                    endgame,   //Flag, if game is running endgame=false
                    error,     //Flag for input validation
                    canPlay,   //Flag if player has a playable card
                    trnOver;   //Flag, if turn is over=true
    Scores          *scores;   //Hold end game scores
*****************************************************************************

      DO
        PRINT "How many players? (2 to 10 Players):"
        CALL clear()
            CALL ignore()
        SET error to false
        GET players;
        IF players is less than or equal to 10 and greater than or equal to 2
            SET error to false
        ELSE
            PRINT "Invalid!"
            SET error to true
    WHILE error is true and cin stream is unsuccessful

    CALL filDeck()
    RETURN deck
    CALL shuffle(deck,discard)
    CALL deal(deck, players)
    RETURN hands
    CALL ignore

    FOR every player
        DO
            CREATE name
            PRINT "Enter Player 's Name: "
            GET name
            IF size of name is greater than 25
                PRINT "Error! Name must be less than 25 characters"
                CALL clear()
                CALL ignore()
                SET error to true
            SET player in each hand to name
         WHILE error is true

    FOR every card in deck
        SET cards in draw to cards in deck
    FOR every card in draw
        DESTROY cards in deck
```

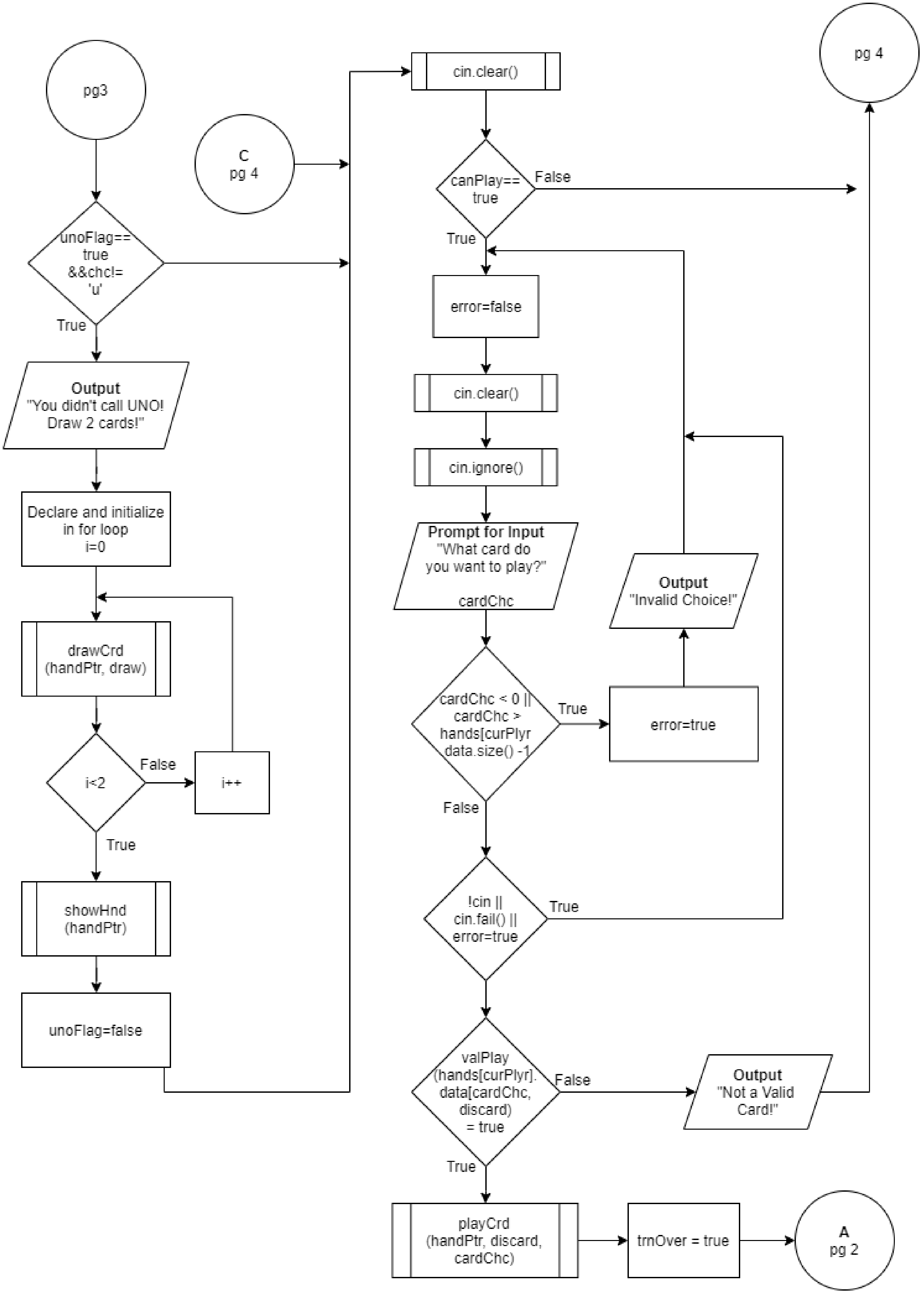
```
DO
    DO
        INCREMENT numHnds
              SET hndSzs at curPlyr to size of hands at curPlyr
        SET handPtr to data in hands at curPlyr
        CALL prcCard(handPtr, draw, discard)
        PRINT player in hands at curPlyr "'s TURN!"
        CALL valPlay(handPtr,discard)
              RETURN canPlay
              CALL uno(handPtr,canPlay)
              RETURN unoFlag
              PRINT "Discard: "
        CALL showCrd(discard)
              CALL showHnd(handPtr)
              PRINT "Enter u To Call UNO or any other key to continue: "
              GET chc
        IF choice is 'u'
            IF unoFlag is true
                PRINT "UNO!!!!"
            ELSE
                IF size of hands at curPlyr is greater than 2
                    PRINT "You still have " size of hands at curPlyr
                          " cards! NO UNO!"
                IF canPlay is false
                    PRINT "You don't have a playable card!"
                          "NO UNO!"
        ELSE IF unoFlag is true and chc is 'u'
                PRINT "You didn't call UNO! Draw 2 cards!"
                FOR 2 iterations
                    CALL drawCrd(handPtr,draw)
                PRINT endline
                CALL showHnd(handPtr)
                SET unoFlag to false
        DO
            CALL clear()
                    IF canPlay is true
                DO
                   SET error to false
                   CALL clear()
                   PRINT "What card do you want to play?"
                   GET  cardChc
                   IF cardChc is less than 0 or cardChc is greater than size
                       of data in hands at curPlyr or input stream is
                           unsuccessful
                       SET error to true
                       PRINT "Invalid Choice!"
                WHILE error is true or input stream is unsuccessful
                          CALL valPlay(hands at curPlyr, discard)
                IF  RETURN is true
                                CALL playCrd(handPtr, discard, cardChc)
                                SET trnOver to true
                ELSE
                    PRINT "Not a Valid Card!"
                    SET trnOver to false;
            ELSE
                DO
                                PRINT "No Valid Card to Play!"
                    PRINT Enter any key to continue"
                    GET cont
                    CALL drawCrd(handPtr, draw)
                                CALL valPlay(handPtr, discard)
                    RETURN canPlay
                    IF canPlay is true
```

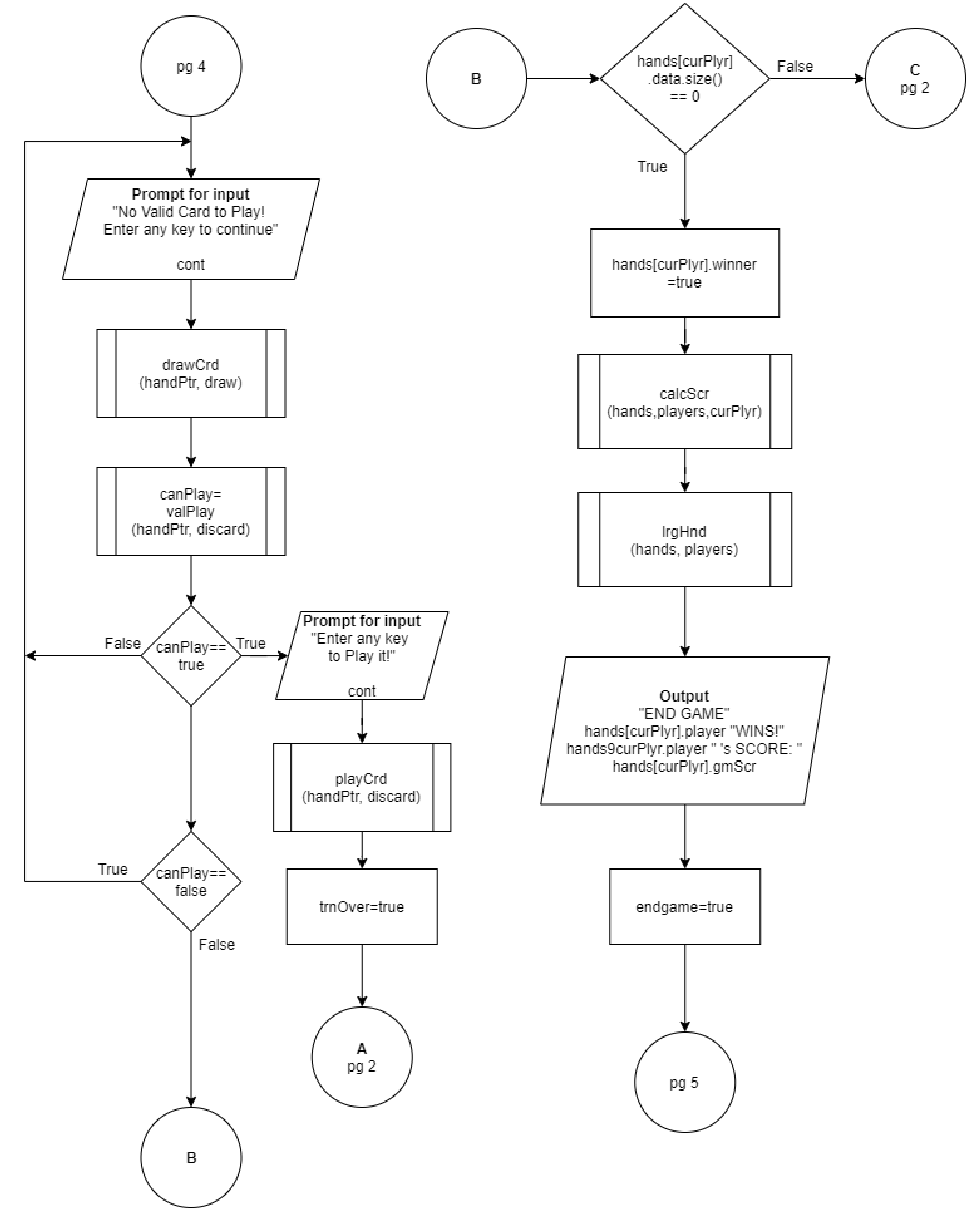
```
                        PRINT "Enter any key to Play it"
                        GET cont
                        CALL playCrd(handPtr, discard)
                        SET trnOver to true
                WHILE canPlay is false
            IF size of data in hands at curPlyr is 0
                          SET winner in hands at curPlyr to true
                          CALL calcScr(hands,players,curPlyr)
                          CALL lrgHnd(hands, players)
                          PRINT end game message
                          PRINT player in hands at curPlyr
                          PRINT gmScr in hands at curPlyr
                SET endgame to true;
        WHILE trnOver is false
        IF size of deck is less than 10
            CALL deck=filDeck()
            CALL shuffle(deck,discard)
     WHILE error is true
    CALL setPlyr(discard, curPlyr, players)
        RETURN curPlyr
 WHILE endgame is false

CALL fillScrs(hands, players)
RETURN scores

CALL wrtBin (scores, players)

DESTROY hands
```

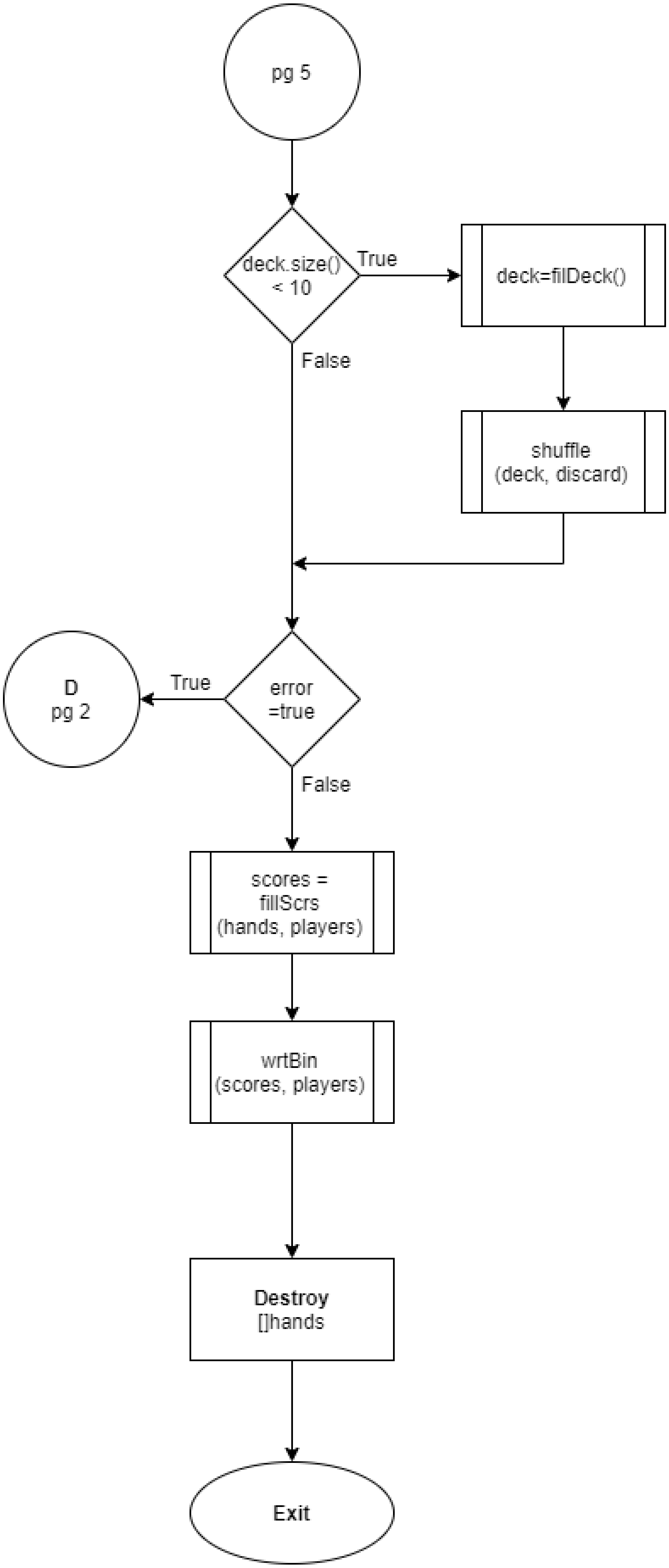
3.2.2 play() Flow Chart:



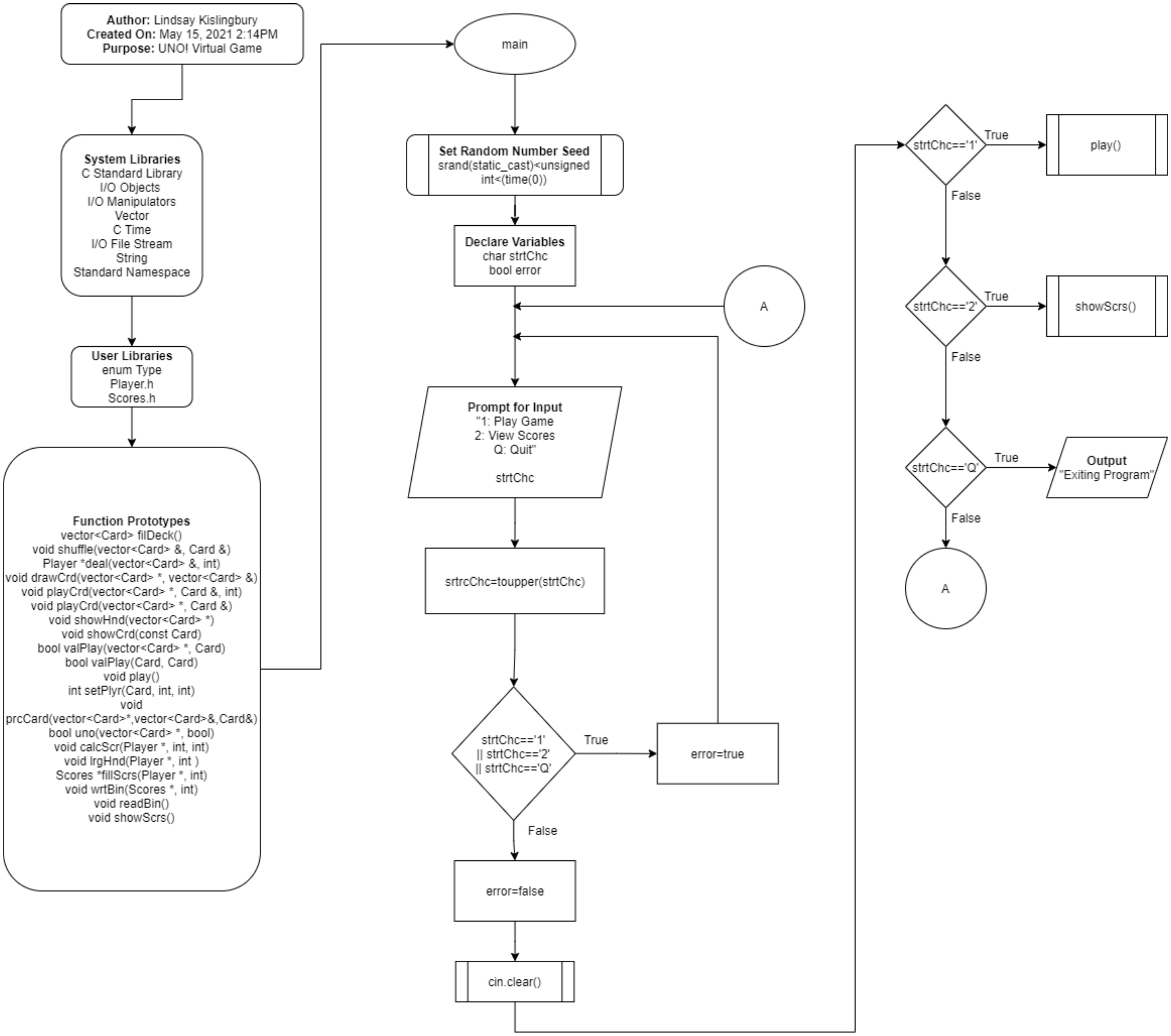

pg 2
Declare and Initialize in for loop i=0
draw.push_back(deck[i])
i<deck.size()
False
i++
True
Declare and Initialize in for loop int i
deck.pop_back()
i<draw.size()
False
i++
True
D pg 5
A pg 3
hands[curPlyr].numHnds++
hands[curPlyr.handSzs.push_back (hands[curPlyr.data.size())
handPtr=&(hands[curPlyr].data)
prcCard (handPtr, draw, discard)
Output hands[curPlyr].player " 's TURN!"
canPlay= valPlay(handPtr,discard)
unoFlag= uno(handPtr,canPlay)
Output "Discard: "
showCrd(discard)
showHnd(handPtr)
Prompt For Input "Enter u To Call UNO or any other key to continue: " chc
chc=='u'
True
False
unoFlag ==true
True
False
Output "UNO!!!"
hands [curPlyr] .data.size() >2
True
False
Output "You still have" hands[curPlyr].data.size() "cards! NO UNO!"
canPlay== false
True
False
Output "You don't have a playable card! NO UNO!"
pg3


pg3
unoFlag== true &&chc!= 'u'
True
Output
"You didn't call UNO! Draw 2 cards!"
Declare and initialize in for loop i=0
drawCrd (handPtr, draw)
i<2
False
i++
True
showHnd (handPtr)
unoFlag=false
C
pg 4
cin.clear()
canPlay== true
False
True
error=false
cin.clear()
cin.ignore()
Prompt for Input
"What card do you want to play?"
cardChc
cardChc < 0 || cardChc > hands[curPlyr data.size() -1
True
error=true
Output
"Invalid Choice!"
False
!cin || cin.fail() || error=true
True
valPlay (hands[curPlyr]. data[cardChc, discard) = true
False
Output
"Not a Valid Card!"
True
playCrd (handPtr, discard, cardChc)
trnOver = true
A
pg 2
pg 4


pg 4
Prompt for input
"No Valid Card to Play!
Enter any key to continue"
cont
drawCrd
(handPtr, draw)
canPlay=
valPlay
(handPtr, discard)
False
canPlay==
true
True
Prompt for input
"Enter any key
to Play it!"
cont
playCrd
(handPtr, discard)
trnOver=true
A
pg 2
True
canPlay==
false
False
B
B
hands[curPlyr]
.data.size()
== 0
False
C
pg 2
True
hands[curPlyr].winner
=true
calcScr
(hands,players,curPlyr)
lrgHnd
(hands, players)
Output
"END GAME"
hands[curPlyr].player "WINS!"
hands9curPlyr.player " 's SCORE: "
hands[curPlyr].gmScr
endgame=true
pg 5


pg 5
deck.size() < 10
True
False
deck=filDeck()
shuffle (deck, discard)
error =true
True
D pg 2
False
scores = fillScrs (hands, players)
wrtBin (scores, players)
Destroy []hands
Exit

3.3 main() Flow Chart:



4 Project Check-Off Sheet

CSC/CIS 17A Project 1 Check-Off Sheet

Chapter	Section	Concept	Points for	Location in	Comments
			Inclusion	Code	
9		**Pointers/Memory Allocation**			
	1	Memory Addresses		163	Assign the memory address of the current player's hand to handPtr, a pointer variable of type vector<Card>
	2	Pointer Variables	5	93	Declare a pointer to a vector<Card>
	3	Arrays/Pointers	5	662	Loop through an array of Player structures passed to the function by pointer
	4	Pointer Arithmetic			
	5	Pointer Initialization		163	Assign the memory address of the current player's hand to handPtr, a pointer variable of type vector<Card>
	6	Comparing			
	7	Function Parameters	5	516	A pointer to a vector<Card> variable used as a parameter for the valPlay function
	8	Memory Allocation	5	353	Dynamically allocate memory to an array of Player structures
	9	Return Parameters	5	364	Return a dynamically allocated array of vector<Card> from function deal
	10	Smart Pointers			
10		**Char Arrays and Strings**			
	1	Testing		143	Input validation on length of string input
	2	Case Conversion		571	Pass input variable to tolower member function
	3	C-Strings	10		
	4	Library Functions		142, 143	Use getline and .length() to validate input
	5	Conversion		665	Convert string name to char array
	6	Your own functions			
	7	Strings	10	140	String input
11		**Structured Data**			

	1	Abstract Data Types		Player.h Scores.h enum Type	
	2	Data			
	3	Access		425	showHnd function accesses Card structures by Type
	4	Initialize			
	5	Arrays	5	643	Modify elements in array of Player structures
	6	Nested	5	Player.h	Player structure has an element of vector<Card> Card structure has element which is an enumerated data type Type
	7	Function Arguments	5	221	Pass a Card structure to valPlay function
	8	Function Return	5	685	Return scores structure from fillScrs function
	9	Pointers	5	610	Access Card structure by pointer
	10	Unions ****			
	11	Enumeration	5	21	Enumerated data type Type declared

12		**Binary Files**			
	1	File Operations			
	2	Formatting	2		
	3	Function Parameters	2		
	4	Error Testing		721	Continue reading until the end of the file is reached
	5	Member Functions	2		
	6	Multiple Files	2		
	7	Binary Files	5	689	This function writes to a binary file
	8	Records with Structures	5	696	Write the members of Scores record to binary file
	9	Random Access Files	5		
	10	Input/Output Simultaneous	2		
		Total	100		