Project 1
<Uno Game V9>

CSC-17a
Miguel Galvez
November 14,2021

Introduction

Title: Uno Game

Uno is a standard card game in which players must match the color or value of the card on the pile and toss their matching card accordingly.

The winner is the player who has 0 cards left in their hands. In standard Uno, people must shout "Uno" when they are left with one card in their hand.

Otherwise, if someone calls them out for not shouting "Uno", they must draw a card.

However, in my version of Uno, there are only two players and there is no shouting requirement. The players must simply add to the pile until one person no longer has any cards.

Summary

Project Size: 513 lines

The number of variables:17

My version of Uno is fully operational in a 2-player format. However, as a result of there only being 2 players, the reverse card results in being a useless card since the reverse order of play still leads to the next player in queue. However, in a future object-oriented version. I'll probably add more players so that the reverse card will have that functionality.

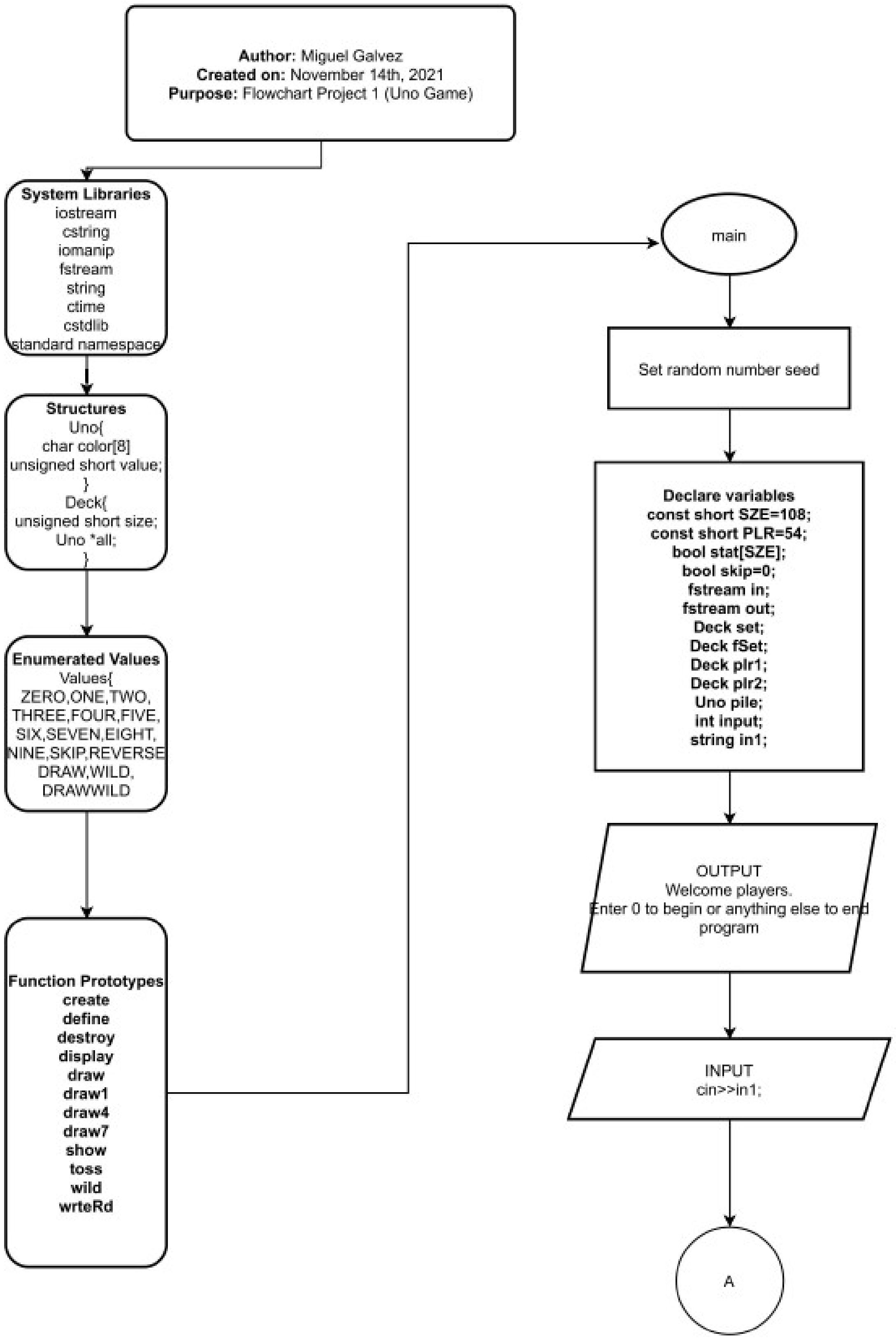
I programmed utilizing most of the concepts I learned from this class regarding dynamic memory,binary files, strings/cstrings, and structures/enumeration.

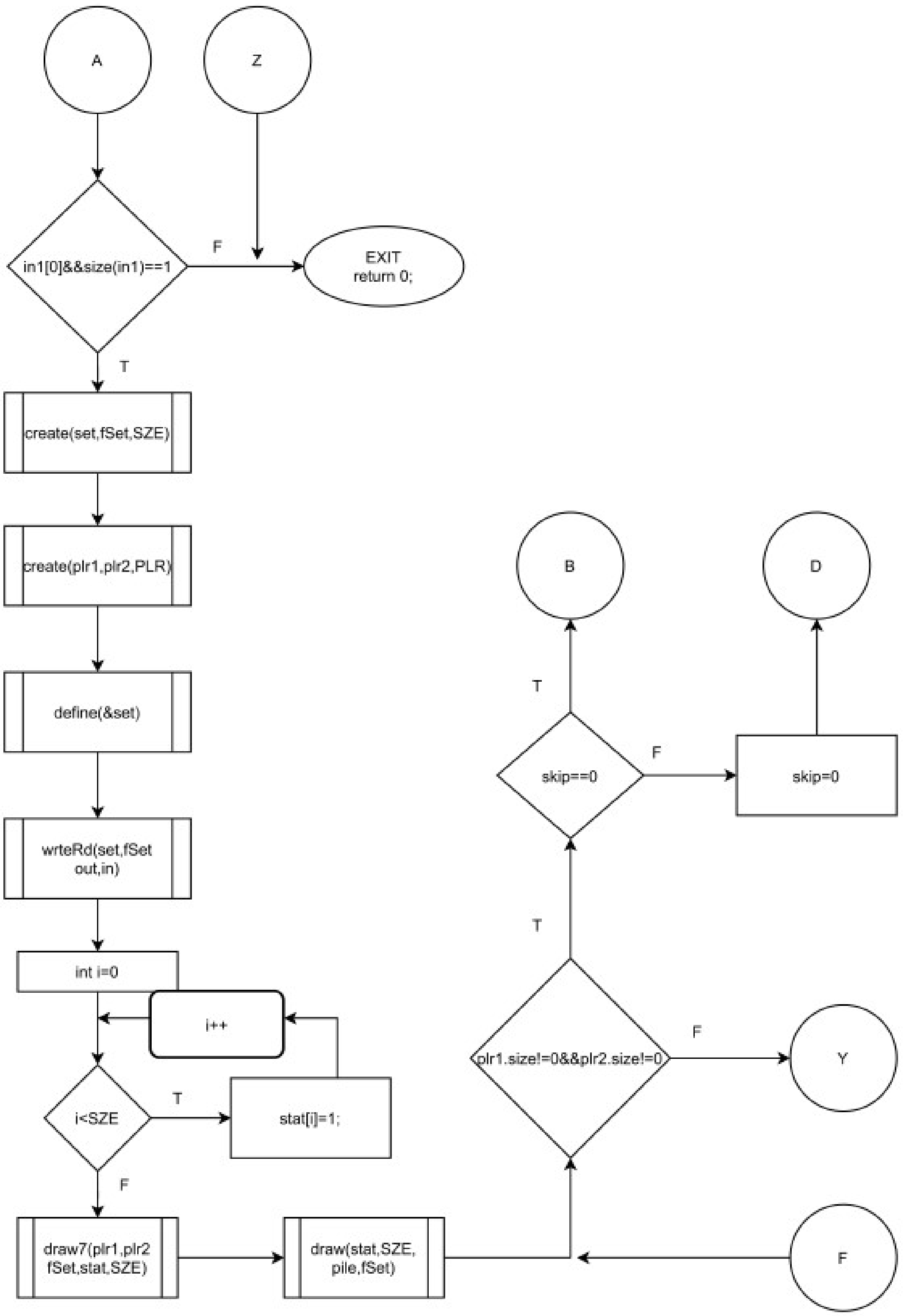
The project took me about 3 weeks to complete. The first week I struggled to write a structure containing pointers to a file. So I redid version 1 into version 2 so that there is only one pointer in my nested structure. In the rest of the following weeks, I had to figure out the logic behind Uno rules and semantics. Overall, I enjoyed the experience I gained from this project.

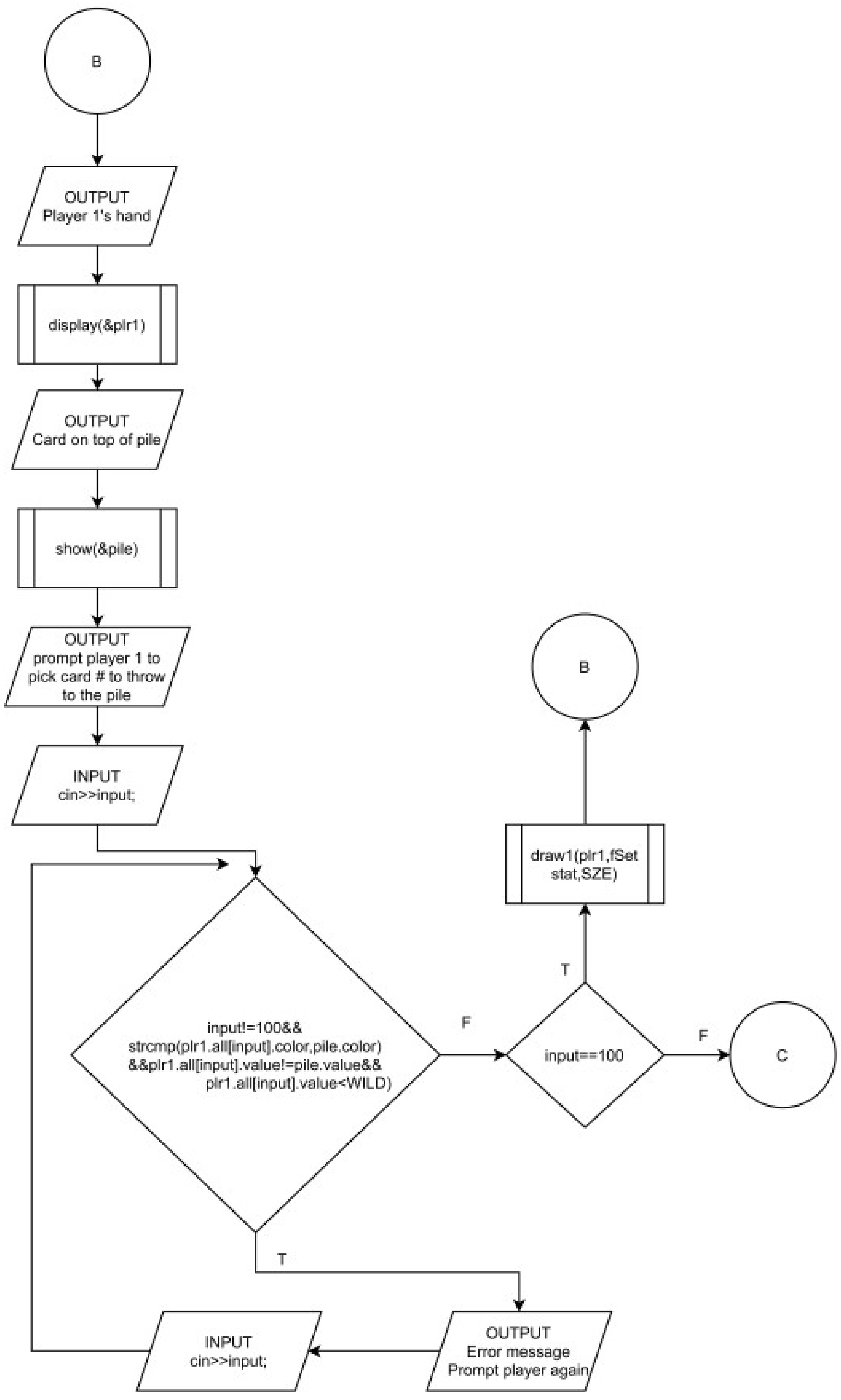
Description

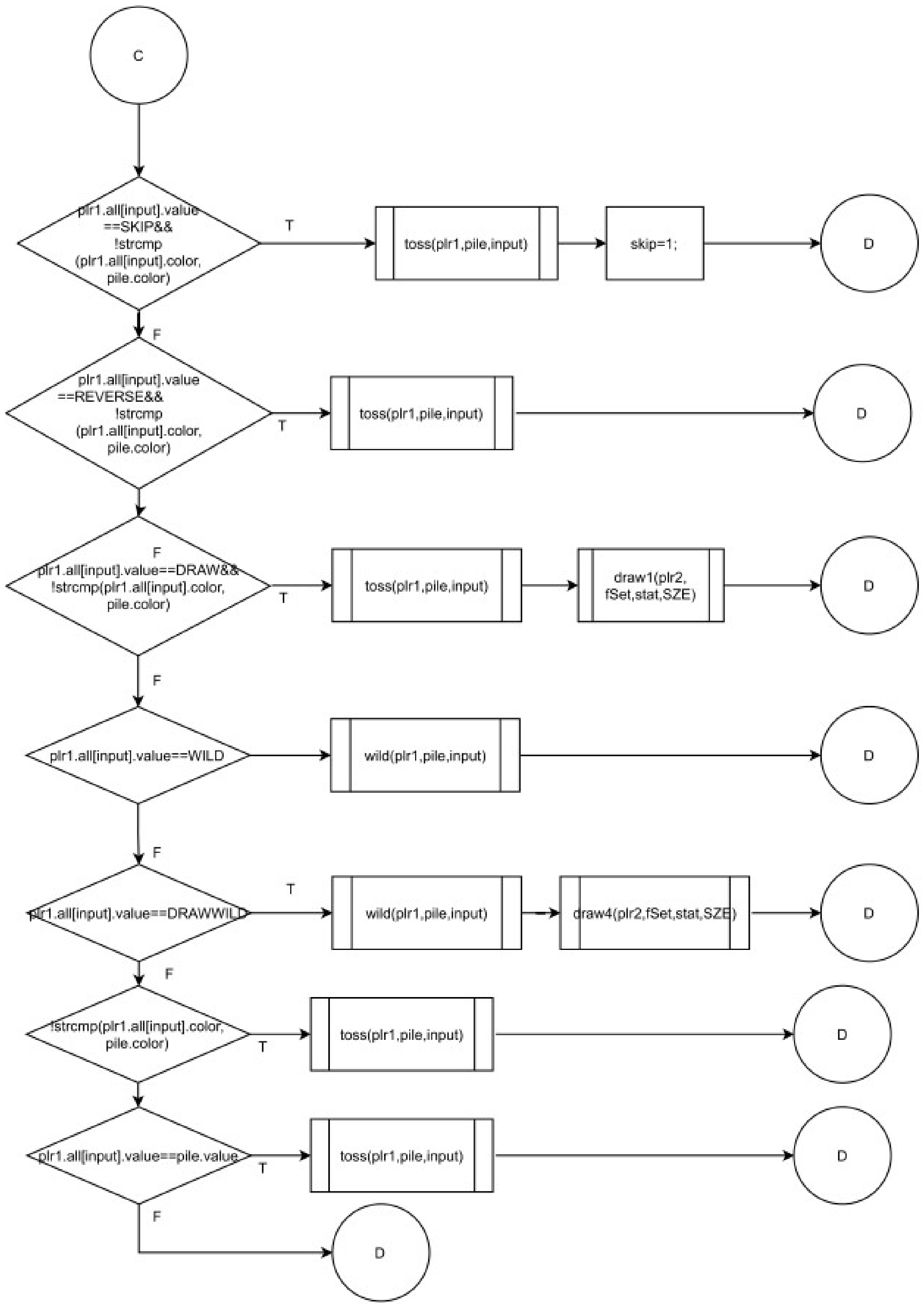
The objective of this program is to incorporate all that was learned from CSC5 review.

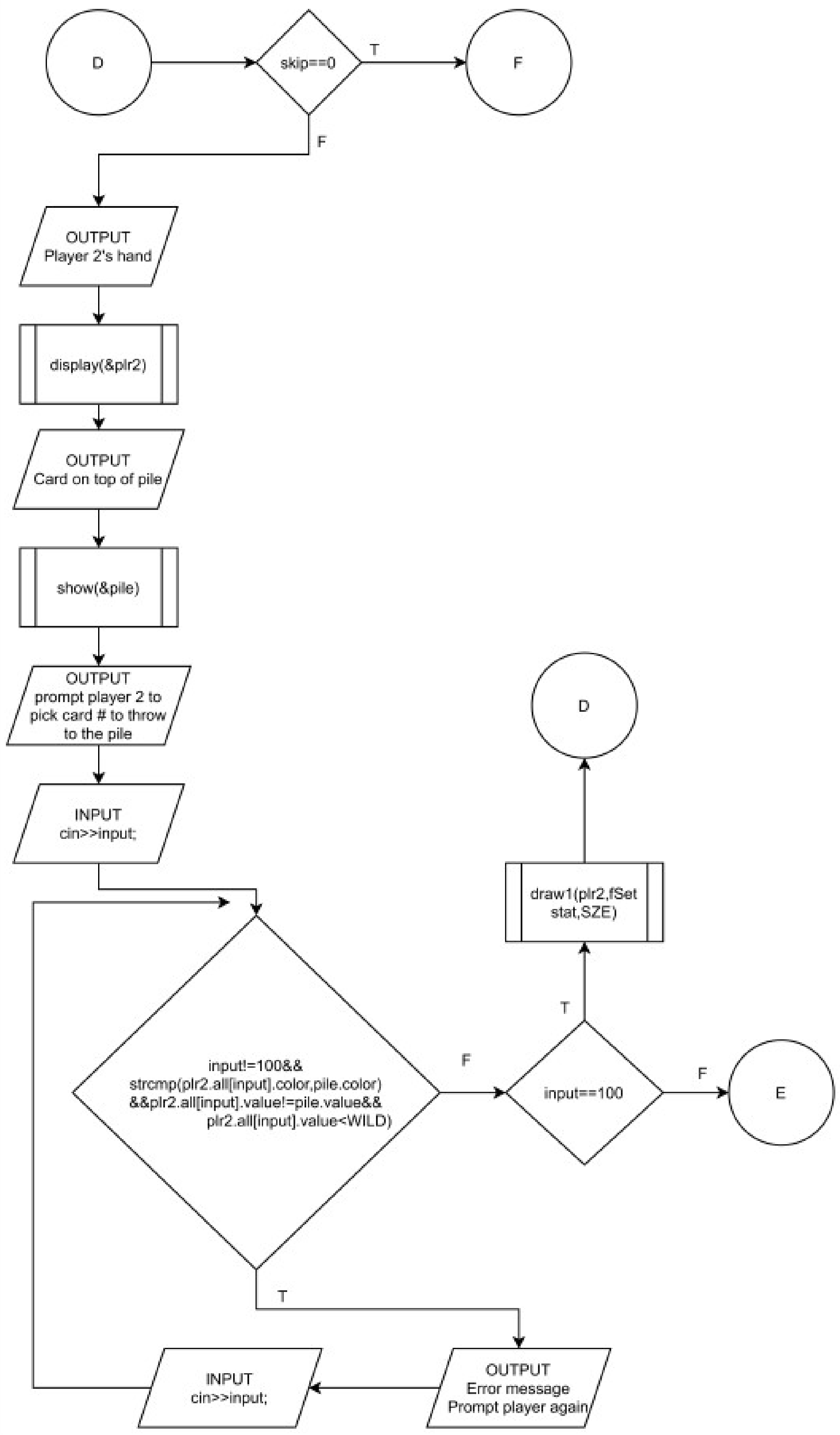
Flowchart

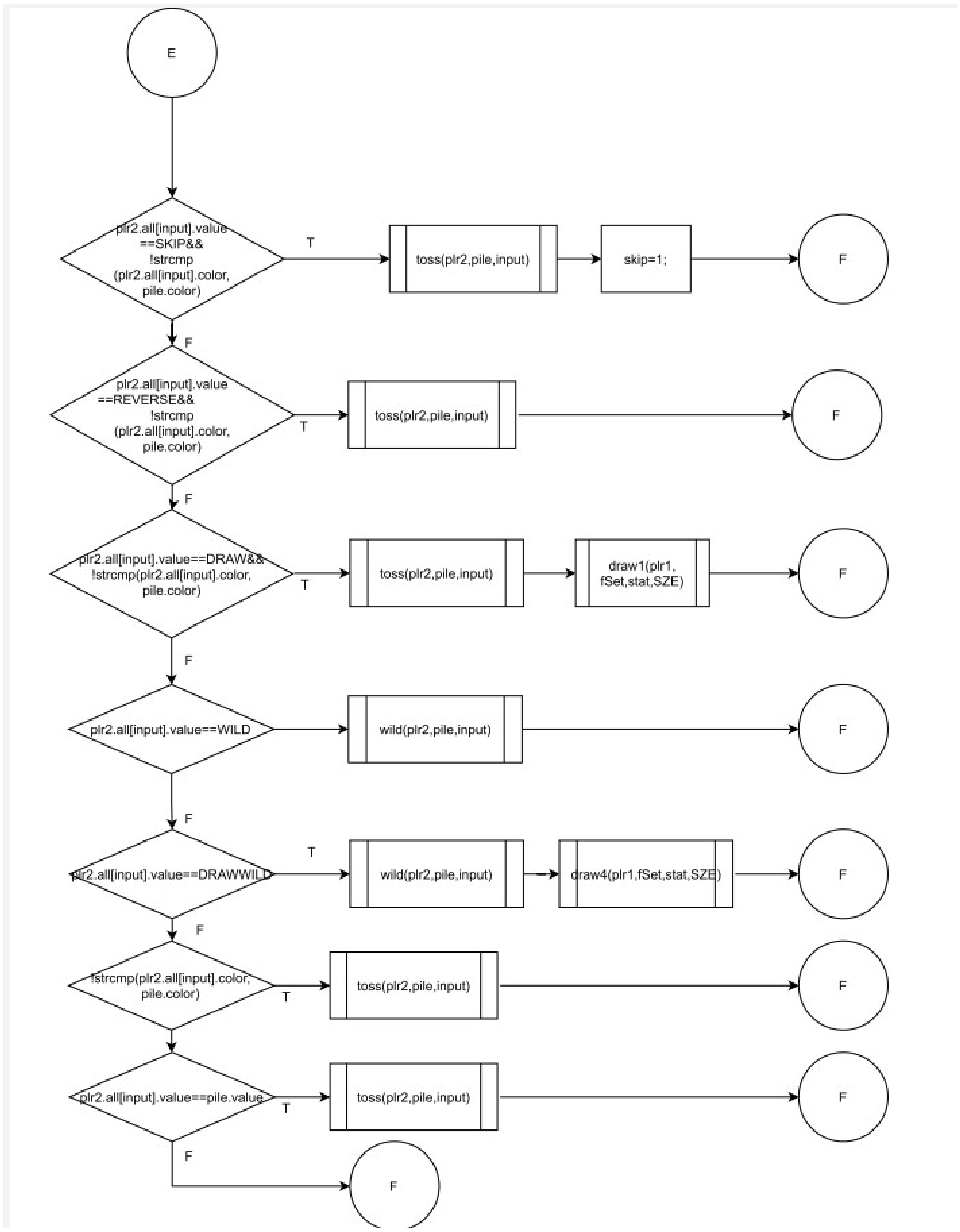

Author: Miguel Galvez
Created on: November 14th, 2021
Purpose: Flowchart Project 1 (Uno Game)
System Libraries
iostream
cstring
iomanip
fstream
string
ctime
cstdlib
standard namespace
Structures
Uno{
char color[8]
unsigned short value;
}
Deck{
unsigned short size;
Uno *all;
}
Enumerated Values
Values{
ZERO,ONE,TWO,
THREE,FOUR,FIVE,
SIX,SEVEN,EIGHT,
NINE,SKIP,REVERSE
DRAW,WILD,
DRAWWILD
Function Prototypes
create
define
destroy
display
draw
draw1
draw4
draw7
show
toss
wild
wrteRd
main
Set random number seed
Declare variables
const short SZE=108;
const short PLR=54;
bool stat[SZE];
bool skip=0;
fstream in;
fstream out;
Deck set;
Deck fSet;
Deck plr1;
Deck plr2;
Uno pile;
int input;
string in1;
OUTPUT
Welcome players.
Enter 0 to begin or anything else to end program
INPUT
cin>>in1;
A

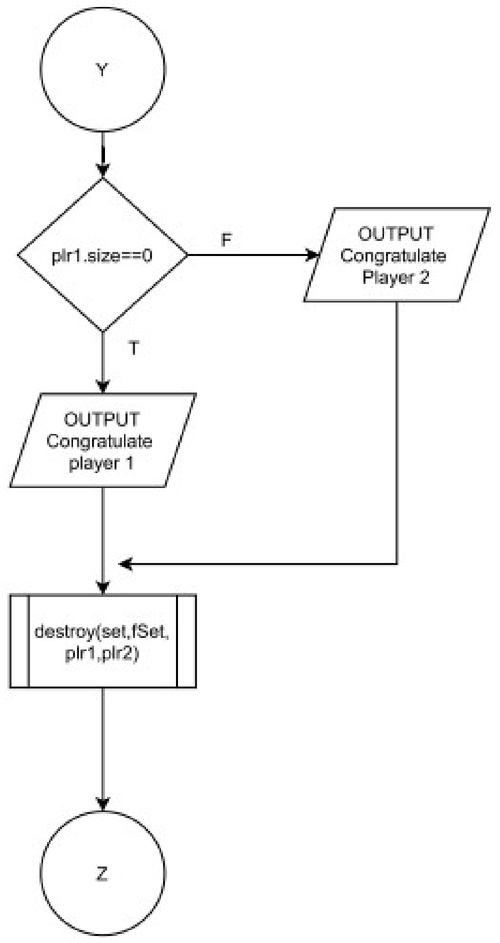

A
Z
in1[0]&&size(in1)==1
F
EXIT
return 0;
T
create(set,fSet,SZE)
create(plr1,plr2,PLR)
define(&set)
wrteRd(set,fSet
out,in)
int i=0
i++
i<SZE
T
stat[i]=1;
F
draw7(plr1,plr2
fSet,stat,SZE)
draw(stat,SZE,
pile,fSet)
F
plr1.size!=0&&plr2.size!=0
F
Y
T
skip==0
F
skip=0
T
B
D


B
OUTPUT
Player 1's hand
display(&plr1)
OUTPUT
Card on top of pile
show(&pile)
OUTPUT
prompt player 1 to
pick card # to throw
to the pile
INPUT
cin>>input;
input!=100&&
strcmp(plr1.all[input].color,pile.color)
&&plr1.all[input].value!=pile.value&&
plr1.all[input].value<WILD)
F
input==100
T
draw1(plr1,fSet
stat,SZE)
B
F
C
T
OUTPUT
Error message
Prompt player again
INPUT
cin>>input;


C
plr1.all[input].value ==SKIP&& !strcmp (plr1.all[input].color, pile.color)
T
toss(plr1,pile,input)
skip=1;
D
F
plr1.all[input].value ==REVERSE&& !strcmp (plr1.all[input].color, pile.color)
T
toss(plr1,pile,input)
D
F
plr1.all[input].value==DRAW&& !strcmp(plr1.all[input].color, pile.color)
T
toss(plr1,pile,input)
draw1(plr2, fSet,stat,SZE)
D
F
plr1.all[input].value==WILD
wild(plr1,pile,input)
D
F
plr1.all[input].value==DRAWWILD
T
wild(plr1,pile,input)
draw4(plr2,fSet,stat,SZE)
D
F
!strcmp(plr1.all[input].color, pile.color)
T
toss(plr1,pile,input)
D
plr1.all[input].value==pile.value
T
toss(plr1,pile,input)
D
F
D


D
skip==0
T
F
F
OUTPUT Player 2's hand
display(&plr2)
OUTPUT Card on top of pile
show(&pile)
OUTPUT prompt player 2 to pick card # to throw to the pile
INPUT cin>>input;
input!=100&& strcmp(plr2.all[input].color,pile.color) &&plr2.all[input].value!=pile.value&& plr2.all[input].value<WILD)
F
input==100
T
draw1(plr2,fSet stat,SZE)
D
F
E
T
OUTPUT Error message Prompt player again
INPUT cin>>input;


E
plr2.all[input].value ==SKIP&& !strcmp (plr2.all[input].color, pile.color)
T
toss(plr2,pile,input)
skip=1;
F
F
plr2.all[input].value ==REVERSE&& !strcmp (plr2.all[input].color, pile.color)
T
toss(plr2,pile,input)
F
F
plr2.all[input].value==DRAW&& !strcmp(plr2.all[input].color, pile.color)
T
toss(plr2,pile,input)
draw1(plr1, fSet,stat,SZE)
F
F
plr2.all[input].value==WILD
wild(plr2,pile,input)
F
F
plr2.all[input].value==DRAWWILD
T
wild(plr2,pile,input)
draw4(plr1,fSet,stat,SZE)
F
F
!strcmp(plr2.all[input].color, pile.color)
T
toss(plr2,pile,input)
F
plr2.all[input].value==pile.value
T
toss(plr2,pile,input)
F
F
F


Y
plr1.size==0
F
T
OUTPUT Congratulate Player 2
OUTPUT Congratulate player 1
destroy(set,fSet, plr1,plr2)
Z

Pseudo code

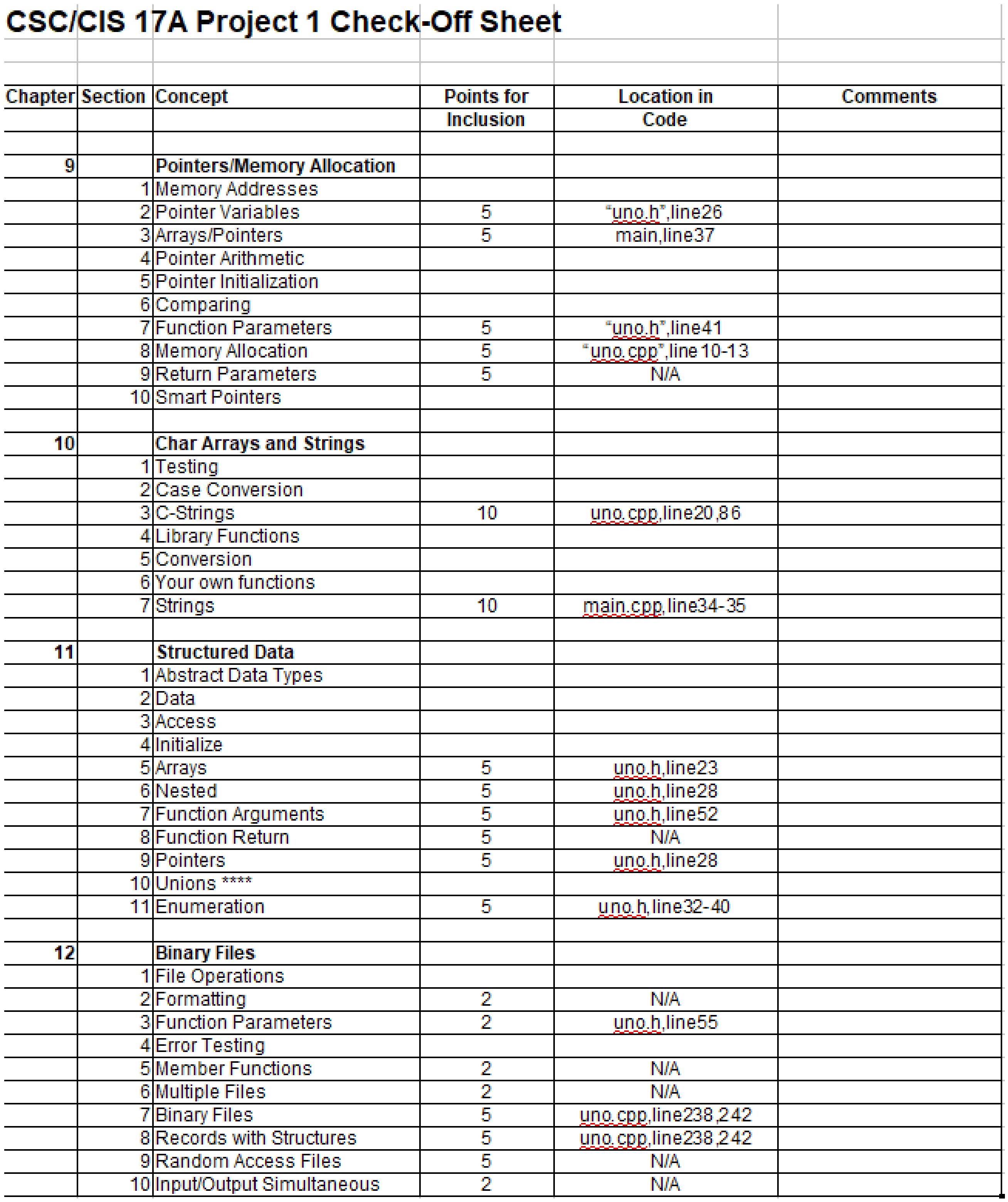
```
start program
enter 0 to start program, or anything else to quit

If user entered 0,
  Start the game
  Create memory for set, set from file(fSet), player 1(plr1) and player 2(plr2)
  Define Uno Cards into set
  Create a binary file from set and write all Uno cards into file
  Read from binary file just created into fSet
  Make bool array stat[SZE] all equal 1
  Draw 7 cards for each player from fSet
  Draw 1 card for pile
  While player 1's hand does not equal 0 and player 2's hand does not equal 0
    If skip equals 0
      Display player 1's hand and card on pile
      Prompt player to select a card number to throw to pile or 100 to draw
      While input is not 100 and colors do not match with pile and value does not
      match with pile and player's card does not equal a wildcard
        Reprompt player 1 for input
      While input is 100
        Draw a card for player 1
        Reprompt player 1 for input
      While input is not 100 and colors do not match with pile and value does not
      match with pile and player's card does not equal a wildcard
        Reprompt player 1 for input
       If player 1 entered a card that's a skip and same color as pile
        toss the card onto the pile
        skip equals 1
       Else If player 1 entered a card that's a reverse and same color as pile
        toss the card onto the pile
       Else If player 1 entered a card that's a Draw and same color as pile
        toss the card onto the pile
        player 2 draws a card
       Else If player 1 entered a card that's a wildcard
        toss the card onto the pile
        pick a color for the pile
       Else If player 1 entered a card that's a wildcard draw +4
        toss the card onto the pile
        pick a color for the pile
        player 2 draws 4 cards
    Else
       skip=0
    If skip equals 0
      Display player 2's hand and card on pile
      Prompt player 2 to select a card number to throw to pile or 100 to draw
      While input is not 100 and colors do not match with pile and value does not
      match with pile and player's card does not equal a wildcard
        Reprompt player 2 for input
      While input is 100
        Draw a card for player 2
        Reprompt player 1 for input
      While input is not 100 and colors do not match with pile and value does not
      match with pile and player's card does not equal a wildcard
        Reprompt player 2 for input
       If player 2 entered a card that's a skip and same color as pile
        toss the card onto the pile
        skip equals 1
       Else If player 2 entered a card that's a reverse and same color as pile
        toss the card onto the pile
       Else If player 2 entered a card that's a Draw and same color as pile
        toss the card onto the pile
        player 2 draws a card
       Else If player 2 entered a card that's a wildcard
        toss the card onto the pile
        pick a color for the pile
       Else If player 2 entered a card that's a wildcard draw +4
        toss the card onto the pile
        pick a color for the pile
        player 1 draws 4 cards
    Else
       skip=0
  If player 1's hand size is 0
    Congratulate player 1
  Else
    Congratulate player 2
  Cleanup/Deallocate Memory from structures
end program
```

Major Variables

Type	**Variable Name**	**Description**	**Location**
Deck	set	Set of Uno cards	Main(),line23
Deck	fSet	set of uno cards from file	main(),line24
Deck	plr1	player 1's hand	main(),line25
Deck	plr2	player 2's hand	main(),line26
Uno	pile	card on the pile	main(),line27
fstream	in	input file	main(),line21
fstream	out	output file	main(),line 22
const short	SZE	Uno deck size	main(),line 19
const short	PLR	Player hand size	main(),line38
bool	stat[SZE]	Status of what cards are available in the deck	main(),line 19,48,50
bool	skip	status of whether following player needs to be skipped or not	main(),line 52,126,128,202
int	input	primary source of input from players	main(),lines 28,63,65,67,71
string	in1	beginning input from user whether to initialize or end the program	main(),line29,34,35
char	color[8]	color associated with uno card	uno.h,line23 (inside Uno structure)
unsigned short	value	value associated with uno card	uno.h line24 (inside Uno structure)
unsigned short	size	size of Uno deck	uno.h line 27 (inside Deck structure)
Uno	*all	Pointer to data containing all uno	uno.h line 28 (inside Deck structure)

		cards in deck	

Checkoff list



CSC/CIS 17A Project 1 Check-Off Sheet

Chapter	Section	Concept	Points for Inclusion	Location in Code	Comments
9		**Pointers/Memory Allocation**			
	1	Memory Addresses			
	2	Pointer Variables	5	"uno.h",line26	
	3	Arrays/Pointers	5	main,line37	
	4	Pointer Arithmetic			
	5	Pointer Initialization			
	6	Comparing			
	7	Function Parameters	5	"uno.h",line41	
	8	Memory Allocation	5	"uno.cpp",line 10-13	
	9	Return Parameters	5	N/A	
	10	Smart Pointers			
10		**Char Arrays and Strings**			
	1	Testing			
	2	Case Conversion			
	3	C-Strings	10	uno.cpp,line20,86	
	4	Library Functions			
	5	Conversion			
	6	Your own functions			
	7	Strings	10	main.cpp,line34-35	
11		**Structured Data**			
	1	Abstract Data Types			
	2	Data			
	3	Access			
	4	Initialize			
	5	Arrays	5	uno.h,line23	
	6	Nested	5	uno.h,line28	
	7	Function Arguments	5	uno.h,line52	
	8	Function Return	5	N/A	
	9	Pointers	5	uno.h,line28	
	10	Unions ****			
	11	Enumeration	5	uno.h,line32-40	
12		**Binary Files**			
	1	File Operations			
	2	Formatting	2	N/A	
	3	Function Parameters	2	uno.h,line55	
	4	Error Testing			
	5	Member Functions	2	N/A	
	6	Multiple Files	2	N/A	
	7	Binary Files	5	uno.cpp,line238,242	
	8	Records with Structures	5	uno.cpp,line238,242	
	9	Random Access Files	5	N/A	
	10	Input/Output Simultaneous	2	N/A	

References

1. Tony Gaddis, *Starting out with C++ From Control Structures through Objects [7th Ed]*
2. Official UNO Rules (officialgamerules.org)
3. How many cards in uno? A complete breakdown of each card (unorules.org)

Program

```
/*
 * File:   uno.h
 * Author: Miguel Galvez
 * Purpose: Store all function prototypes/libraries/enums
 * Created on November 13, 2021, 3:14 PM
 */

#ifndef UNO_H
#define UNO_H

#include <iostream> //cin,cout,endl
#include <cstring>  //strcpy, strcmp
#include <iomanip>  //xprecision,fixed,showpoint
#include <fstream>  //file operations
#include <string>   //string class
#include <ctime>    //time()
#include <cstdlib>  //rand()
#include <valarray>
using namespace std;
//User Libraries Here
//Structures
struct Uno{               //Deck of Uno Cards
    char color[8];        //Color associated with uno card
    unsigned short value;   //Value associated with the card
};
struct Deck{       //Structure of structure containing deck of uno cards
    unsigned short size;       //Size of Uno Deck
    Uno *all;                  //Data of all uno cards
};
//Global Constants Only, No Global Variables
//Like PI, e, Gravity, or conversions
enum Values{       //Card Values
    ZERO,ONE,TWO,THREE,FOUR,    //0-4
    FIVE,SIX,SEVEN,EIGHT,NINE,  //5-9
    SKIP,          //Skip Next Player's Turn
    REVERSE,       //Reverse The Turn Order
    DRAW,         //Next Person Draws a Card
    WILD,         //Wildcard (Pick a color for pile)
    DRAWWILD      //Wildcard Draw +4 Cards
};
//Function Prototypes Here
void create(Deck&,Deck&,short); //Dynamically allocate memory
void define(Deck *);          //Define Uno cards
void destroy(Deck&,Deck&,Deck&,Deck&);  //Cleanup
void display(Deck *);         //Display hand
void draw(bool [],short,Uno &,Deck);   //Draw a single card to pile
void draw1(Deck&,Deck,bool[],short);   //Draw a single card to hand
void draw4(Deck&,Deck,bool[],short);   //Draw four cards to hands
void draw7(Deck&,Deck&,Deck,          //Draw seven cards to hands
        bool [],short);
void show(Uno *);             //Show the top of pile
void toss(Deck&,Uno&,int);     //Toss card from player's hand to pile
void wild(Deck&,Uno&,int);     //Handle case for wildcard
void wrteRd(Deck &set,Deck &fSet,//Write entire deck of uno cards to file
      fstream &out,fstream &in);//and read file into another deck
#endif /* UNO_H */
/*
 * File:   main.cpp
 * Author: Miguel Galvez
 * Created on November 13th 2021, 7:22PM
 * Purpose: Lastly, we will move function prototypes
 *          /structures/libraries to our header file and function
 *          declarations to their own .cpp file
 */

//System Libraries Here
#include "uno.h"
//Program Execution Begins Here
int main(int argc, char** argv) {
    //Set Random Number Generator Seed
    srand(static_cast<unsigned int>(time(0)));
    //Declare all Variables Here
    const short SZE=108;   //Uno Deck Size
    const short PLR=54;    //Player hand size
    bool stat[SZE]; //Status of Deck array
    bool skip=0;   //Skip flag
    fstream in;    //input file
    fstream out;   //output file
```

```
Deck set;       //Deck of uno cards
Deck fSet;      //Deck of uno cards from file
Deck plr1;      //Player 1
Deck plr2;      //Player 2
Uno pile;       //Current card on the pile
int input;      //user input
string in1;     //First user input
//Display Uno game header
cout<<"Welcome to the game of Uno! "<<endl
   <<"Enter 0 to begin, or anything else to "
   <<"end this program. ";
cin>>in1;
if(in1[0]==48&&size(in1)==1){
   //Dynamically Allocate Memory for Deck and Players
   create(set,fSet,SZE);
   create(plr1,plr2,PLR);
   //Players start off with 0 cards before drawing
   //Define Uno Cards for Deck
   define(&set);
   //Write and read deck of cards
   wrteRd(set,fSet,out,in);
         //Default bool values to true
         for(int i=0;i<SZE;i++)
            stat[i]=1;
         //Draw cards from fSet into Players
         draw7(plr1,plr2,fSet,stat,SZE);
         //Draw a card for pile
         draw(stat,SZE,pile,fSet);
         while(plr1.size!=0&&plr2.size!=0){
            if(skip==0){
               //Show player 1's hands
               cout<<"Player 1's hand: "<<endl;
               display(&plr1);
               //Display card on top of pile
               cout<<"Card on \ntop \nof pile: ";
               show(&pile);
               //Prompt player 1 for an applicable card to throw to the pile
               //Otherwise draw a card
               cout<<"Enter the card # to throw to the pile. "<<endl
                  <<"Otherwise, enter 100 to draw: ";
               cin>>input;
               //Input Validation
```

```
while(input!=100&&strcmp(plr1.all[input].color,pile.color)
   &&plr1.all[input].value!=pile.value&&
   plr1.all[input].value<WILD){
   cout<<"Error: enter a valid number."<<endl;
   cout<<"Enter the card # to throw to the pile. "<<endl
      <<"Otherwise, enter 100 to draw: ";
   cin>>input;
}
//If card is valid, toss card onto pile
while(input==100){
   draw1(plr1,fSet,stat,SZE);
   //Show player 1's hands
   cout<<"Player 1's hand: "<<endl;
   display(&plr1);
   //Display card on top of pile
   cout<<"Card on \ntop \nof pile: ";
   show(&pile);
   cout<<"Enter the card # to throw to the pile. \n"
      <<"Or enter 100 to draw. \n";
   cin>>input;
   //Input Validation
   while(input!=100&&strcmp(plr1.all[input].color,pile.color)
      &&plr1.all[input].value!=pile.value&&
      plr1.all[input].value<WILD){
      cout<<"Error: enter a valid number."<<endl;
      cout<<"Enter the card # to throw to the pile. "<<endl
         <<"Otherwise, enter 100 to draw: ";
      cin>>input;
}
//If value is skip and same color as pile
}if(plr1.all[input].value==SKIP&&
      !strcmp(plr1.all[input].color,pile.color)){
   toss(plr1,pile,input);
   skip=1;     //Set the skip flag
//If value is reverse and same color as pile
}else if(plr1.all[input].value==REVERSE&&
      !strcmp(plr1.all[input].color,pile.color)){
   toss(plr1,pile,input);
//If value is draw +1 and same color as pile
}else if(plr1.all[input].value==DRAW&&
      !strcmp(plr1.all[input].color,pile.color)){
   toss(plr1,pile,input);
```

```
        draw1(plr2,fSet,stat,SZE);
    //If value is a wildcard, toss and pick color for pile
    }else if(plr1.all[input].value==WILD){
        wild(plr1,pile,input);
    //If value is wildcard +4, toss, pick color for pile
    //and make next player draw four cards
    }else if(plr1.all[input].value==DRAWWILD){
        wild(plr1,pile,input);
        //Now plr2 needs to draw 4 cards
        draw4(plr2,fSet,stat,SZE);
    //If color is the same as pile
    }else if(!strcmp(plr1.all[input].color,pile.color)){
        toss(plr1,pile,input);
    //If value is the same as pile
    }else if(plr1.all[input].value==pile.value){
        toss(plr1,pile,input);
    }
//If skip flag is set, skip player's turn and reset flag
}else{
    skip=0;
}
if(skip==0){
    //Display player 2's hand
    cout<<"Player 2's hand: "<<endl;
    display(&plr2);
    //Display top of pile
    cout<<"Card on \ntop \nof pile: ";
    show(&pile);
    //Prompt player 1 for an applicable card to throw
    //to the pile. Otherwise, draw a card
    cout<<"Enter the card # to throw to the pile. "<<endl;
    cin>>input;
    //Input Validation
    while(input!=100&&strcmp(plr2.all[input].color,pile.color)
        &&plr2.all[input].value!=pile.value
        &&plr2.all[input].value<WILD){
        cout<<"Error: enter a valid number."<<endl;
        cout<<"Enter the card # to throw to the pile. \n"
            <<"Or enter 100 to draw. \n";
        cin>>input;
    }
    //If card is valid, toss card onto pile
```

```
          while(input==100){
             draw1(plr2,fSet,stat,SZE);
             //Display player 2's hand
             cout<<"Player 2's hand: "<<endl;
             display(&plr2);
             //Display top of pile
             cout<<"Card on \ntop \nof pile: ";
             show(&pile);
             cout<<"Enter the card # to throw to the pile. \n"
                <<"Or enter 100 to draw. \n";
             cin>>input;
             //Input Validation
             while(input!=100&&strcmp(plr2.all[input].color,pile.color)
                &&plr2.all[input].value!=pile.value
                &&plr2.all[input].value<WILD){
                cout<<"Error: enter a valid number."<<endl;
                cout<<"Enter the card # to throw to the pile. \n"
                   <<"Or enter 100 to draw. \n";
                cin>>input;
             }
          }
          if(plr2.all[input].value==SKIP&&
                !strcmp(plr2.all[input].color,pile.color)){
             toss(plr2,pile,input);
             skip=1;     //Set the skip flag
          //If card is reverse and same color as pile
          }else if(plr2.all[input].value==REVERSE&&
                !strcmp(plr2.all[input].color,pile.color)){
             toss(plr2,pile,input);
          //If card is draw +1 and same color as pile
          }else if(plr2.all[input].value==DRAW&&
                !strcmp(plr2.all[input].color,pile.color)){
             toss(plr2,pile,input);
             //Make player 1 draw a card
             draw1(plr1,fSet,stat,SZE);
          //If card is a wildcard, pick a color for pile
          }else if(plr2.all[input].value==WILD){
             wild(plr2,pile,input);
          //If card is wildcard draw +4, pick a color for pile
          //and next player draws 4 cards
          }else if(plr2.all[input].value==DRAWWILD){
             wild(plr2,pile,input);
```

```
                    //Make player 1 draw four cards
                    draw4(plr1,fSet,stat,SZE);
                //If card is same color as pile
                }else if(!strcmp(plr2.all[input].color,pile.color)){
                    toss(plr2,pile,input);
                //If card is same value as pile
                }else if(plr2.all[input].value==pile.value){
                    toss(plr2,pile,input);
                }
            //If skip flag is set, skip player and reset flag
            }else{
                skip=0;
            }
        }
        //Congratulate Winner
        plr1.size==0?cout<<"Congratulations! Player 1 wins."<<endl:
                     cout<<"Congratulations! Player 1 wins."<<endl;
    //Cleanup
    destroy(set,fSet,plr1,plr2);
    }
    //Exit
    return 0;
}
#include "uno.h"
/*
 * File:   main.cpp
 * Author: Miguel Galvez
 * Created on November 13th 2021, 3:46PM
 * Purpose: All function declarations are here
 */

void create(Deck &set,Deck &fSet,short SZE){
    set.all=new Uno [SZE];
    set.size=SZE;
    fSet.all=new Uno [SZE];
    fSet.size=SZE;
}
void define(Deck *x){
    //Define x of Uno cards
    for(int i=0;i<x->size;i++){
        //First 25 cards are Green
        if(i<25){
            strcpy(x->all[i].color, "Green");
            i==0?x->all[i].value=ZERO:
            i<=2?x->all[i].value=ONE:
            i<=4?x->all[i].value=TWO:
            i<=6?x->all[i].value=THREE:
            i<=8?x->all[i].value=FOUR:
            i<=10?x->all[i].value=FIVE:
            i<=12?x->all[i].value=SIX:
            i<=14?x->all[i].value=SEVEN:
            i<=16?x->all[i].value=EIGHT:
            i<=18?x->all[i].value=NINE:
            i<=20?x->all[i].value=SKIP:
            i<=22?x->all[i].value=REVERSE:
            x->all[i].value=DRAW;
        //Next 25 are Blue
        }else if(i<50){
            strcpy(x->all[i].color, "Blue");
            i==25?x->all[i].value=ZERO:
            i<=27?x->all[i].value=ONE:
            i<=29?x->all[i].value=TWO:
            i<=31?x->all[i].value=THREE:
            i<=33?x->all[i].value=FOUR:
            i<=35?x->all[i].value=FIVE:
            i<=37?x->all[i].value=SIX:
            i<=39?x->all[i].value=SEVEN:
            i<=41?x->all[i].value=EIGHT:
            i<=43?x->all[i].value=NINE:
            i<=45?x->all[i].value=SKIP:
            i<=47?x->all[i].value=REVERSE:
            x->all[i].value=DRAW;
        //Next 25 are Red
        }else if(i<75){
            strcpy(x->all[i].color, "Red");
            i==50?x->all[i].value=ZERO:
            i<=52?x->all[i].value=ONE:
            i<=54?x->all[i].value=TWO:
            i<=56?x->all[i].value=THREE:
            i<=58?x->all[i].value=FOUR:
            i<=60?x->all[i].value=FIVE:
            i<=62?x->all[i].value=SIX:
            i<=64?x->all[i].value=SEVEN:
            i<=66?x->all[i].value=EIGHT:
```

```
            i<=68?x->all[i].value=NINE:
            i<=70?x->all[i].value=SKIP:
            i<=72?x->all[i].value=REVERSE:
            x->all[i].value=DRAW;
        //Next 25 are Yellow
        }else if(i<100){
            strcpy(x->all[i].color, "Yellow");
            i==75?x->all[i].value=ZERO:
            i<=77?x->all[i].value=ONE:
            i<=79?x->all[i].value=TWO:
            i<=81?x->all[i].value=THREE:
            i<=83?x->all[i].value=FOUR:
            i<=85?x->all[i].value=FIVE:
            i<=87?x->all[i].value=SIX:
            i<=89?x->all[i].value=SEVEN:
            i<=91?x->all[i].value=EIGHT:
            i<=93?x->all[i].value=NINE:
            i<=95?x->all[i].value=SKIP:
            i<=97?x->all[i].value=REVERSE:
            x->all[i].value=DRAW;
        //Last 8 are Black wildcards
        }else{
            strcpy(x->all[i].color, "Black");
            i<=103?x->all[i].value=WILD:
            x->all[i].value=DRAWWILD;
        }
    }
}
void destroy(Deck &set,Deck &fSet,Deck &plr1,
                Deck &plr2){
    delete [] set.all;
    delete [] fSet.all;
    delete [] plr1.all;
    delete [] plr2.all;
}
void display(Deck *x){
    cout<<left<<setw(9)<<"Card  #:"
            <<setw(10)<<"Color"<<setw(5)
            <<"Value"<<endl;
    for(int i=0;i<x->size;i++){
        cout<<setw(6)<<"Card "<<i<<": "
            <<setw(10)<<x->all[i].color
            <<setw(5);
            x->all[i].value==SKIP?cout<<"SKIP"<<endl:
            x->all[i].value==REVERSE?cout<<"REVERSE"<<endl:
            x->all[i].value==DRAW?cout<<"DRAW +1"<<endl:
            x->all[i].value==WILD?cout<<"WILDCARD"<<endl:
            x->all[i].value==DRAWWILD?cout<<"DRAW+4"<<endl:
            cout<<x->all[i].value<<endl;
    }
}
void draw(bool stat[],short SZE,Uno &pile,Deck fSet){
    //Draw a card for the pile
    unsigned int indx=rand()%SZE;
    //Keep trying random number until unique
    while(!stat[indx])
        indx=rand()%SZE;
    //Copy Contents from index in deck to player 1 hand
        strcpy(pile.color,fSet.all[indx].color);
        pile.value=fSet.all[indx].value;
    //Falsify index in bool array
    stat[indx]=0;
}
void draw1(Deck &plr,Deck fSet,bool stat[],short SZE){
    //Draw four cards into player's hand
    unsigned int indx=rand()%SZE;
    //Keep trying random number until unique
    while(!stat[indx])
        indx=rand()%SZE;
    //Copy Contents from index in deck to player 1 hand
    strcpy(plr.all[plr.size].color,fSet.all[indx].color);
    plr.all[plr.size].value=fSet.all[indx].value;
    //Falsify index in bool array
    stat[indx]=0;
    plr.size++;
}
void draw4(Deck &plr,Deck fSet,bool stat[]
            ,short SZE){
    //Draw four cards into player's hand
    unsigned int indx=rand()%SZE;
    //Increment hand size
        plr.size+=4;
    //Draw four cards and assign them to end of array
    for(int i=plr.size-4;i<plr.size;i++){
      //Keep trying random number until unique
      while(!stat[indx])
         indx=rand()%SZE;
      //Copy Contents from index in deck to player 1 hand
      strcpy(plr.all[i].color,fSet.all[indx].color);
      plr.all[i].value=fSet.all[indx].value;
      //Falsify index in bool array
      stat[indx]=0;
   }
}
void draw7(Deck &plr1,Deck &plr2,Deck fSet,
            bool stat[],short SZE){
   //Players start off with 0 cards
   plr1.size=0;
   unsigned int indx=rand()%SZE;
   for(int i=0;i<7;i++){
      //Keep trying random number until unique
      while(!stat[indx])
         indx=rand()%SZE;
      //Increment hand size
      plr1.size++;
      //Copy Contents from index in deck to player 1 hand
      strcpy(plr1.all[i].color,fSet.all[indx].color);
      plr1.all[i].value=fSet.all[indx].value;
      //Falsify index in bool array
      stat[indx]=0;
   }
   plr2.size=0;
   indx=rand()%SZE;
   for(int i=0;i<7;i++){
      //Keep trying random number until unique
      while(!stat[indx])
         indx=rand()%SZE;
      //Increment hand size
      plr2.size++;
      //Copy Contents from index in deck to player 1 hand
      strcpy(plr2.all[i].color,fSet.all[indx].color);
      plr2.all[i].value=fSet.all[indx].value;
      //Falsify index in bool array
      stat[indx]=0;
   }
}
```

```
void show(Uno *x){
   cout<<setw(10)<<x->color
      <<setw(5);
      x->value==SKIP?cout<<"SKIP"<<endl:
      x->value==REVERSE?cout<<"REVERSE"<<endl:
      x->value==DRAW?cout<<"DRAW +1"<<endl:
      x->value==WILD?cout<<"WILDCARD"<<endl:
      x->value==DRAWWILD?cout<<"DRAW+4"<<endl:
      cout<<x->value<<endl;
}
void toss(Deck &plr,Uno &pile,int input){
   //Assign color from player's hand to pile
   strcpy(pile.color,plr.all[input].color);
   //Assign value from player's hand to pile
   pile.value=plr.all[input].value;
   //Remove card from player's hand
   for(int i=input;i<plr.size;i++){
      strcpy(plr.all[i].color,plr.all[i+1].color);
      plr.all[i].value=plr.all[i+1].value;
   }
   //Decrement plr1's hand size
   plr.size--;
}
void wild(Deck &plr,Uno &pile,int input){
   //Assign value from player's hand to pile
   pile.value=plr.all[input].value;
   //Remove card from player's hand
   for(int i=input;i<plr.size;i++){
      strcpy(plr.all[i].color,plr.all[i+1].color);
      plr.all[i].value=plr.all[i+1].value;
   }
   //Assign a color based off player's decision
   do{
   cout<<"Enter :"<<endl
      <<"1 for yellow\n"
      <<"2 for green\n"
      <<"3 for red\n"
      <<"4 for blue\n";
   cin>>input;
   }while(input>4||input<0);   //Input validation
   input==1?strcpy(pile.color,"Yellow"):
   input==2?strcpy(pile.color,"Green"):
   input==3?strcpy(pile.color,"Red"):
       strcpy(pile.color,"Blue");
   plr.size--;
}
void wrteRd(Deck &set,Deck &fSet,fstream &out,
      fstream &in){
   //Write Uno cards to file
   out.open("unoCards.dat",ios::out|ios::binary);
   out.write(reinterpret_cast<char *>(&set),sizeof(set)*set.size);
   out.close();
   //Read Uno cards to separate variable
   in.open("unoCards.dat",ios::in|ios::binary);
   in.read(reinterpret_cast<char *>(&fSet),sizeof(fSet)*fSet.size);
   in.close();
}
```