Project 2

< Blackjack >

24WINTER CIS-5
Name: Diana Marciniak
Date: 2/11/2024

Introduction

Title: Blackjack

The game of blackjack or 21 is the most popular table game offered in casinos. The basic premise of the game is that you want to have a hand value that is closer to 21 than that of the dealer, without going over 21. In blackjack, the cards are valued as follows:

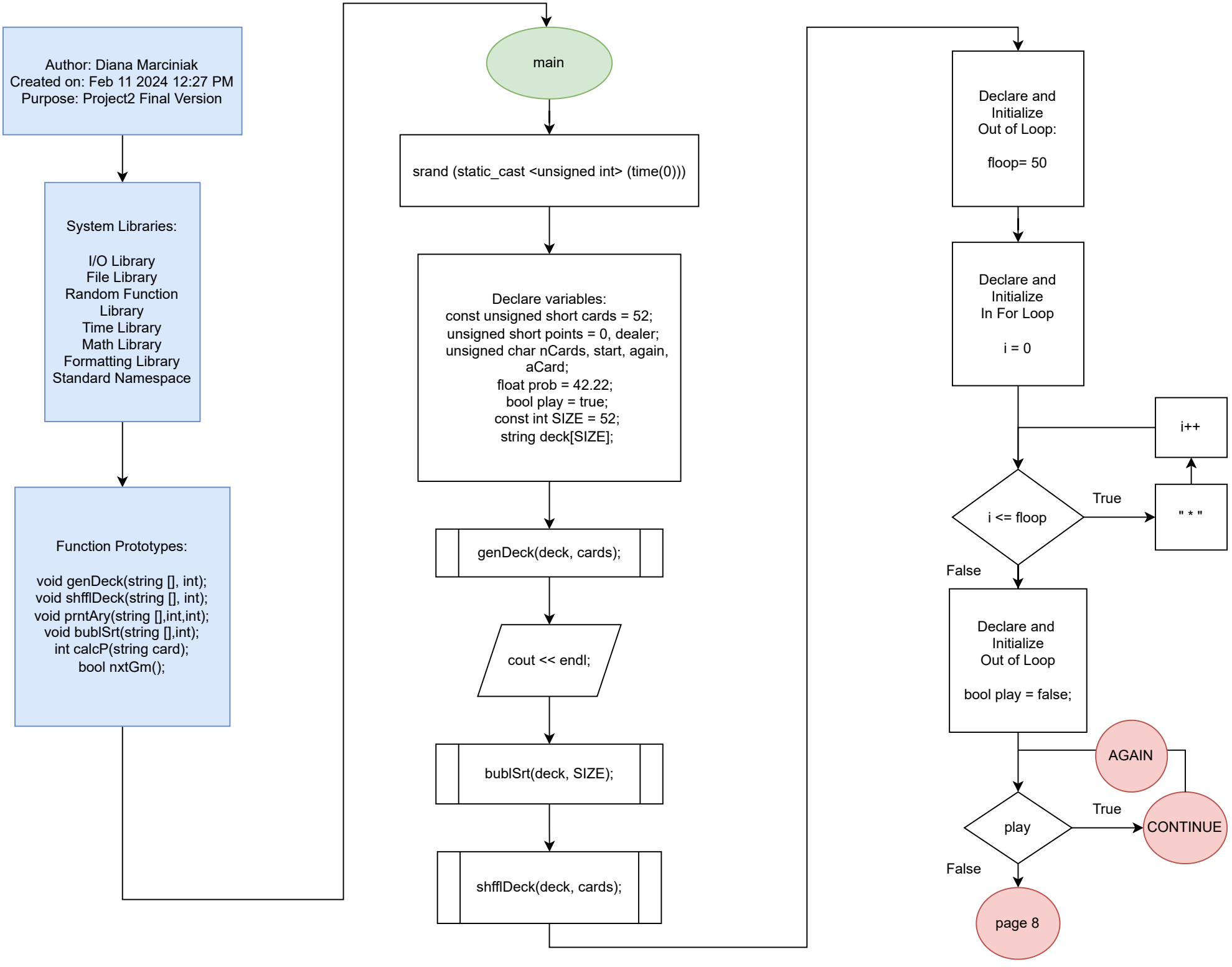
- An Ace can count as either 1 or 11, as explained below (Note: my program counts Ace as 11)
- The cards from 2 through 9 are valued at their face value.
- The 10, Jack, Queen, and King are all valued at 10.

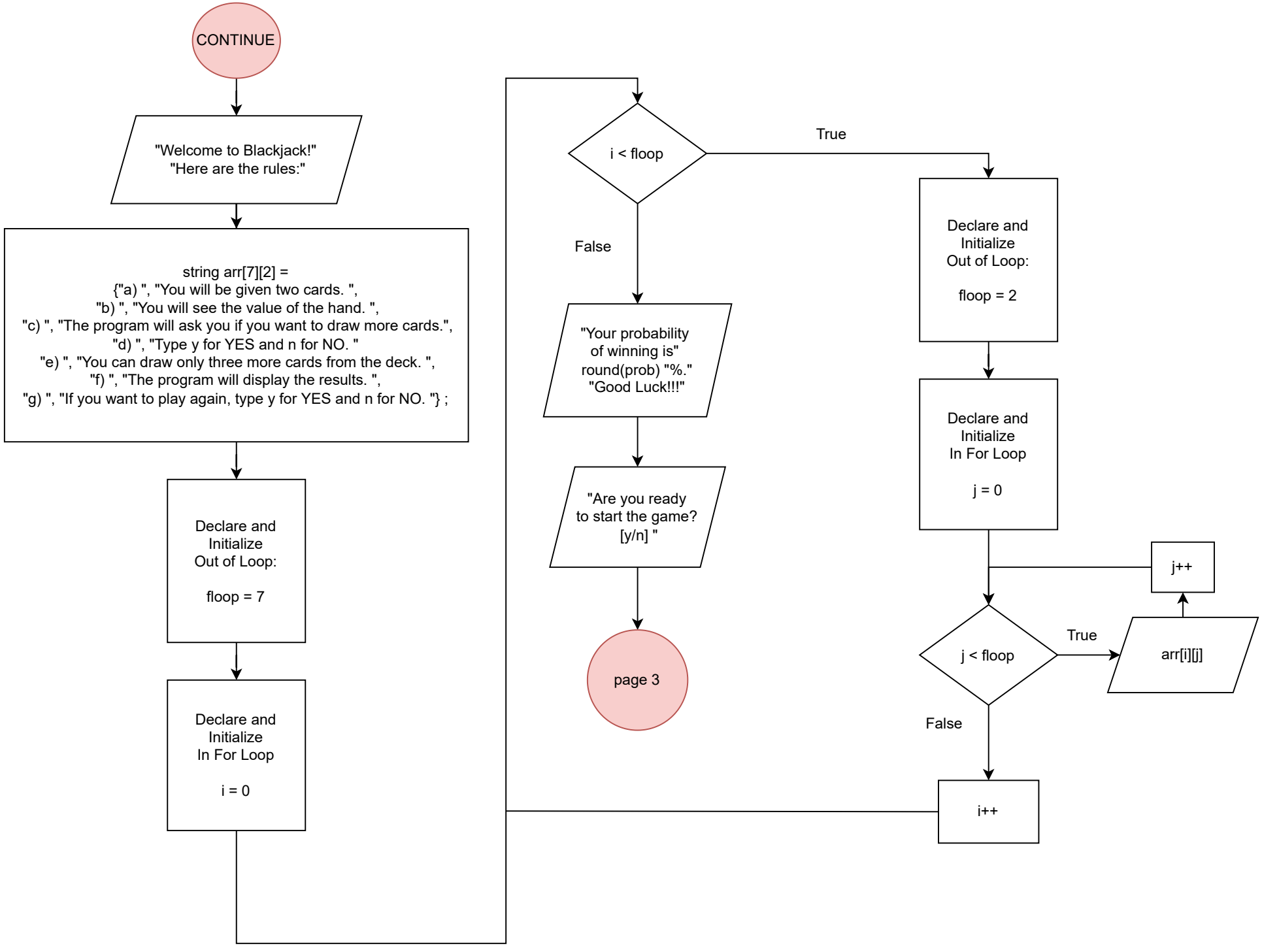
The suits of the cards do not have any meaning in the game. The value of a hand is simply the sum of the point counts of each card in the hand. If you draw a card that makes your hand total go over 21, you loose. A blackjack, or natural, is a total of 21 in your first two cards.

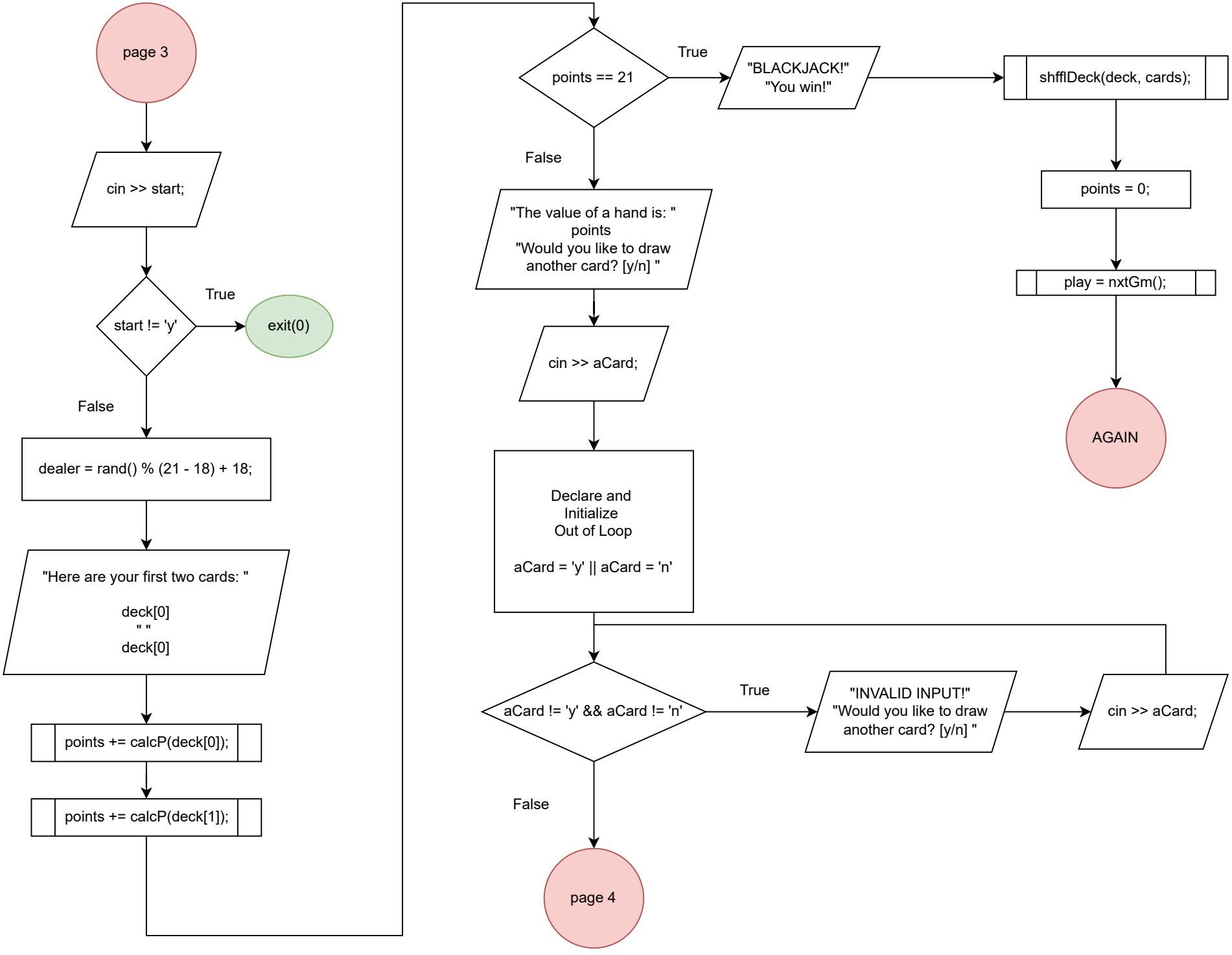
Summary

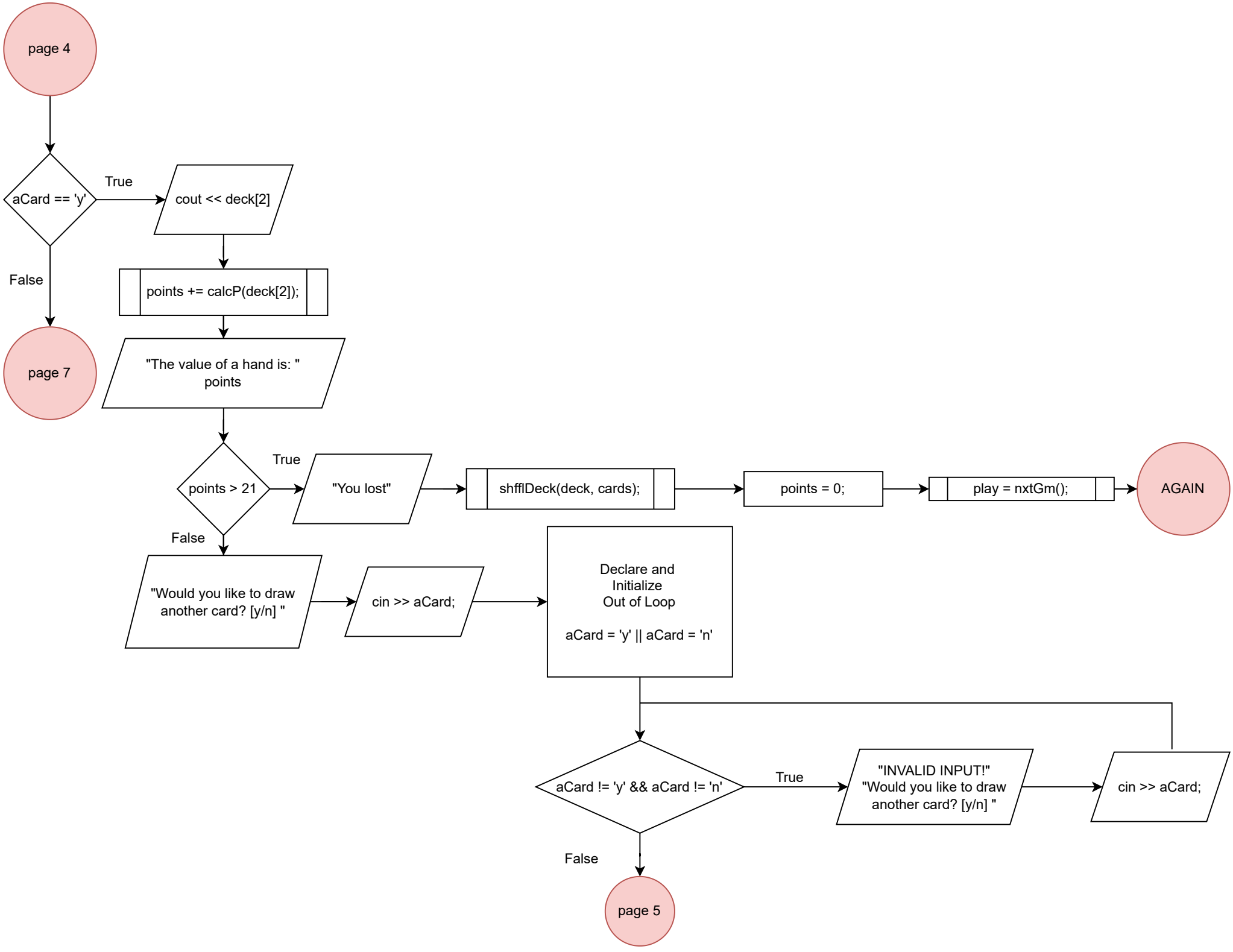
Project size: about 400 lines

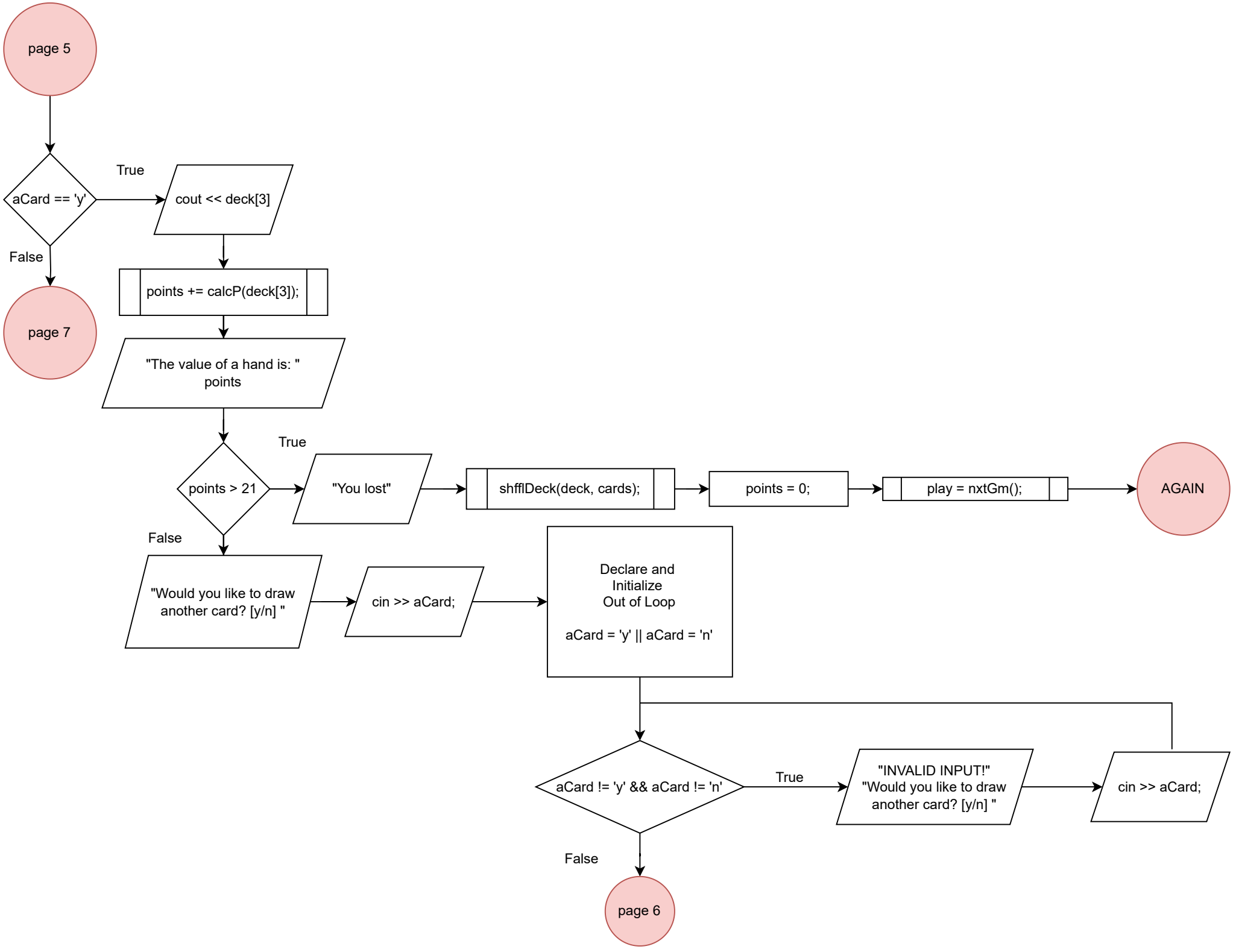
Here is the updated version of a one-player game - blackjack. The rules of the game are displayed at the beginning of the program. The player is given two randomly chosen, unique cards generated by the function void genDeck(string [], int). The program asks the user if he/she wants to draw another card (the user can draw a maximum of 3 more cards). The program calculates the user's points and generates a random number of points for the dealer. The user wins if he/she scores higher than the dealer. Then the is being shuffled and the program asks the user if they want to play the game again, starting with the new cards.

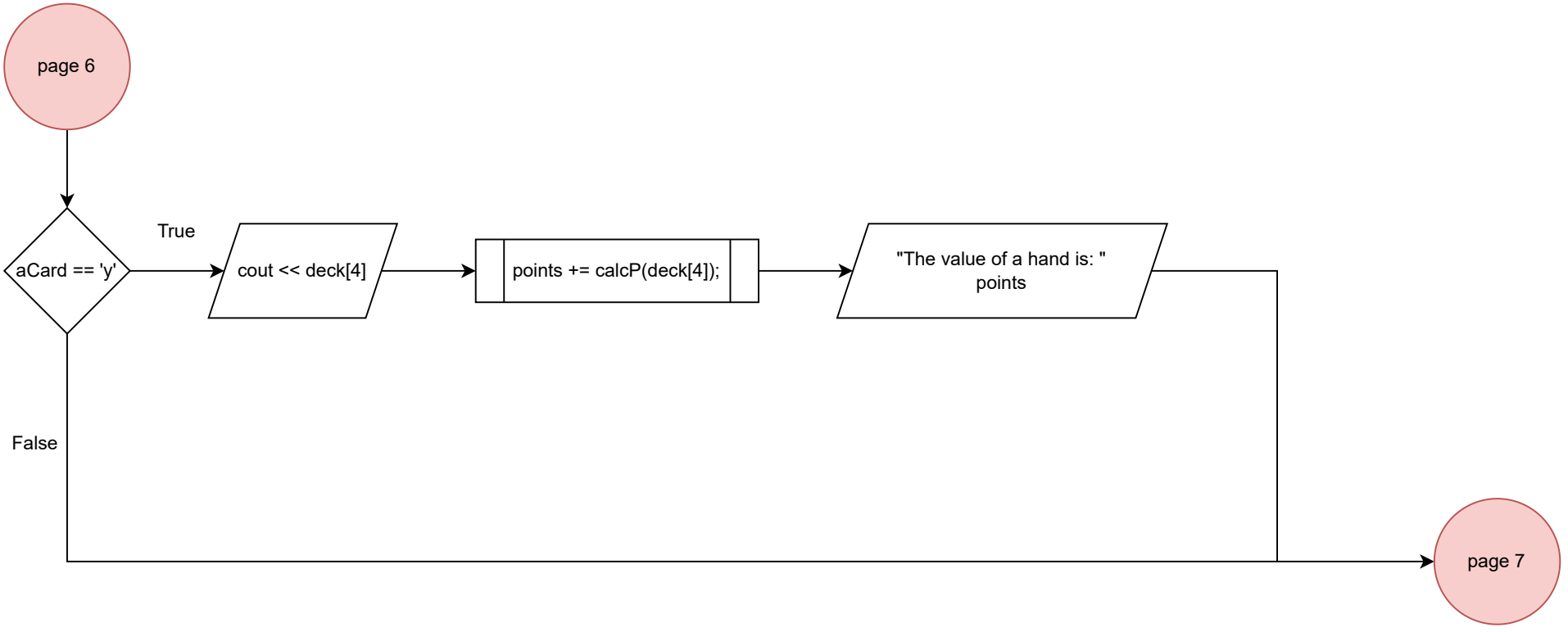

Author: Diana Marciniak
Created on: Feb 11 2024 12:27 PM
Purpose: Project2 Final Version
System Libraries:
I/O Library
File Library
Random Function Library
Time Library
Math Library
Formatting Library
Standard Namespace
Function Prototypes:
void genDeck(string [], int);
void shfflDeck(string [], int);
void prntAry(string [],int,int);
void bublSrt(string [],int);
int calcP(string card);
bool nxtGm();
main
srand (static_cast <unsigned int> (time(0)))
Declare variables:
const unsigned short cards = 52;
unsigned short points = 0, dealer;
unsigned char nCards, start, again, aCard;
float prob = 42.22;
bool play = true;
const int SIZE = 52;
string deck[SIZE];
genDeck(deck, cards);
cout << endl;
bublSrt(deck, SIZE);
shfflDeck(deck, cards);
Declare and Initialize Out of Loop:
floop= 50
Declare and Initialize In For Loop
i = 0
i++
i <= floop
True
" * "
False
Declare and Initialize Out of Loop
bool play = false;
AGAIN
play
True
CONTINUE
False
page 8

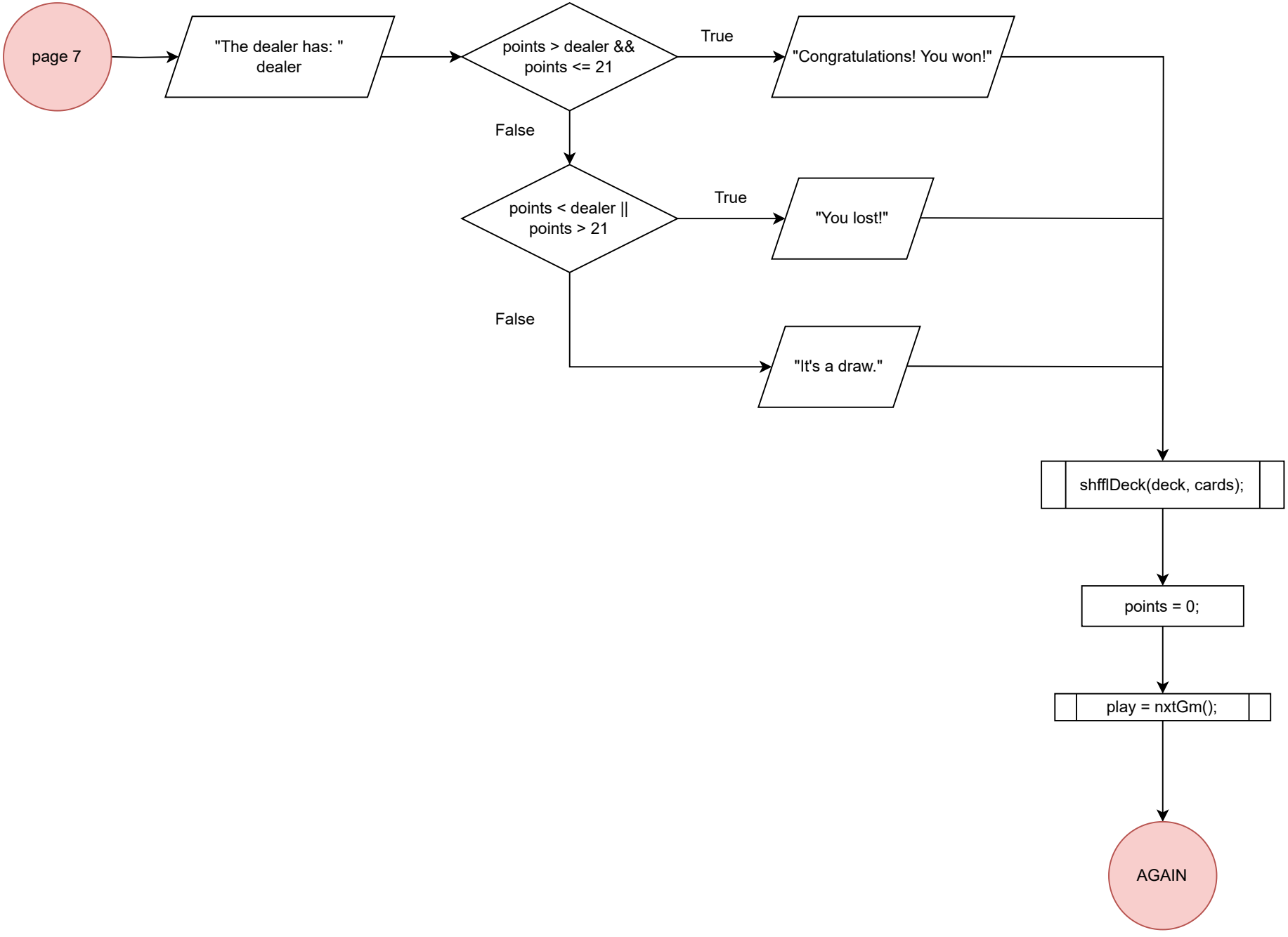

CONTINUE
"Welcome to Blackjack!"
"Here are the rules:"
string arr[7][2] =
{"a) ", "You will be given two cards. ",
"b) ", "You will see the value of the hand. ",
"c) ", "The program will ask you if you want to draw more cards.",
"d) ", "Type y for YES and n for NO. "
"e) ", "You can draw only three more cards from the deck. ",
"f) ", "The program will display the results. ",
"g) ", "If you want to play again, type y for YES and n for NO. "} ;
Declare and Initialize Out of Loop:
floop = 7
Declare and Initialize In For Loop
i = 0
i < floop
True
False
Declare and Initialize Out of Loop:
floop = 2
"Your probability of winning is" round(prob) "%." "Good Luck!!!"
"Are you ready to start the game? [y/n] "
page 3
Declare and Initialize In For Loop
j = 0
j++
j < floop
True
arr[i][j]
False
i++

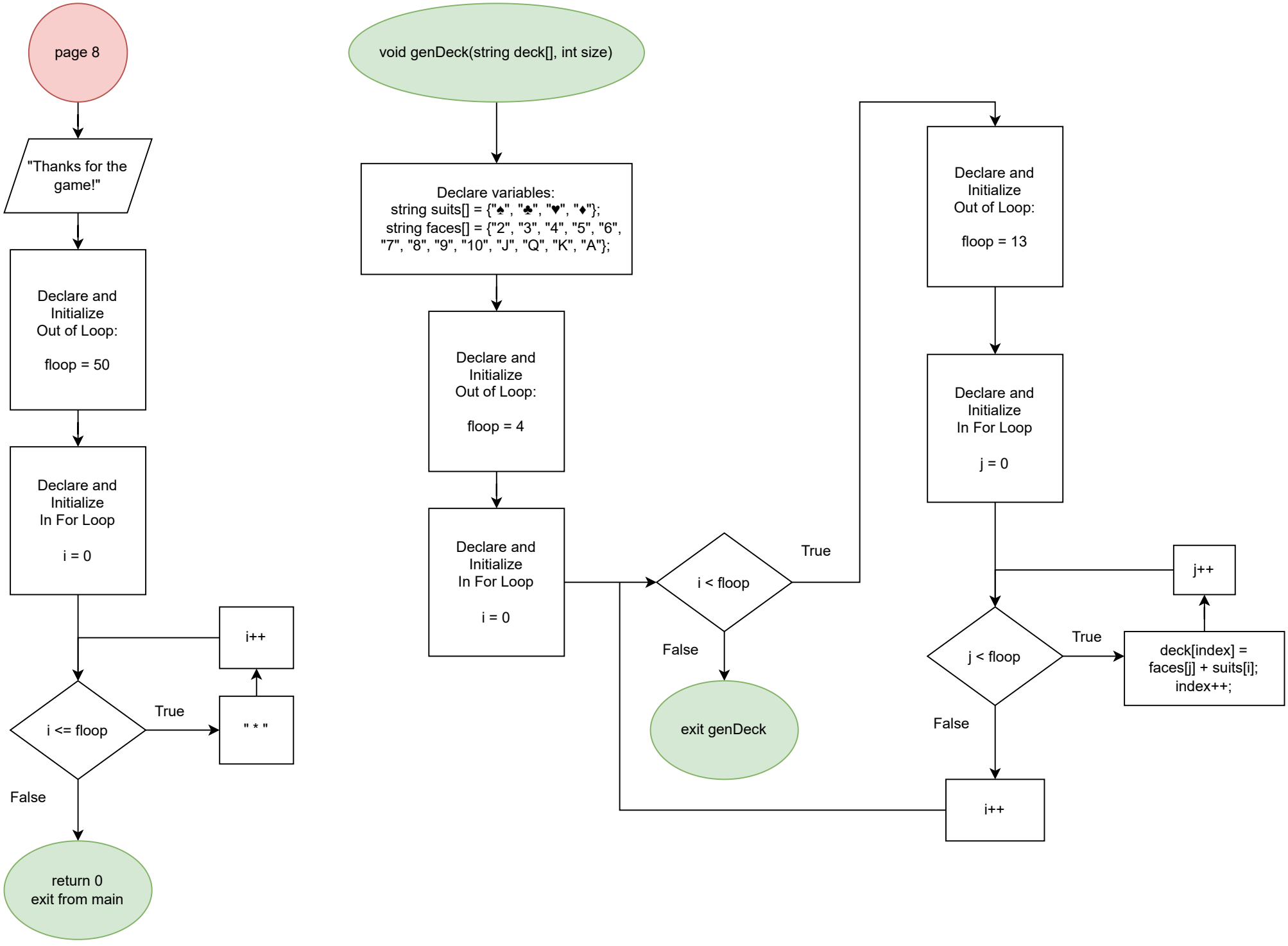

page 3
cin >> start;
start != 'y'
True
exit(0)
False
dealer = rand() % (21 - 18) + 18;
"Here are your first two cards: "
deck[0]
" "
deck[0]
points += calcP(deck[0]);
points += calcP(deck[1]);
points == 21
True
"BLACKJACK!"
"You win!"
shfflDeck(deck, cards);
points = 0;
play = nxtGm();
AGAIN
False
"The value of a hand is: "
points
"Would you like to draw
another card? [y/n] "
cin >> aCard;
Declare and
Initialize
Out of Loop
aCard = 'y' || aCard = 'n'
aCard != 'y' && aCard != 'n'
True
"INVALID INPUT!"
"Would you like to draw
another card? [y/n] "
cin >> aCard;
False
page 4

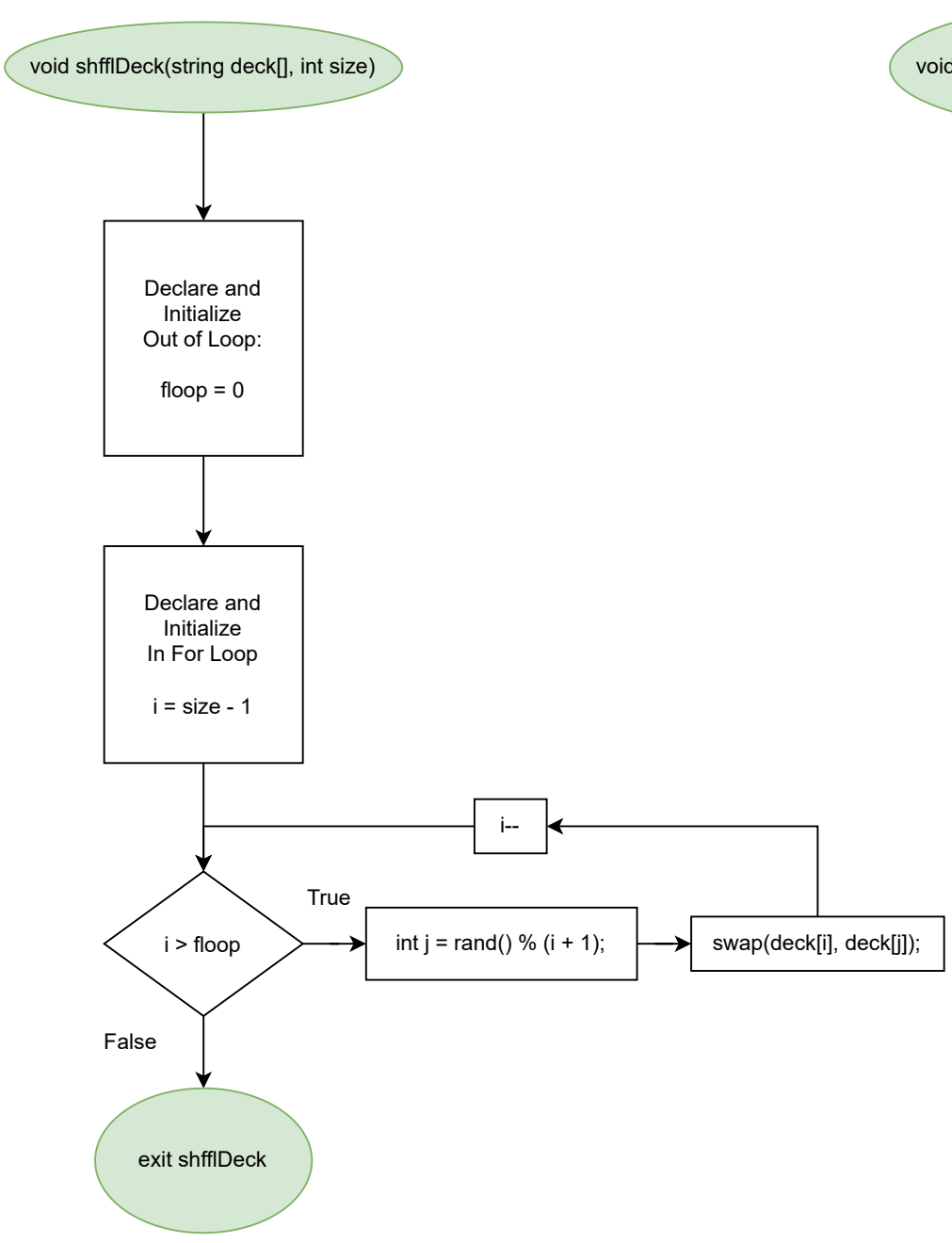

page 4
aCard == 'y'
True
cout << deck[2]
False
points += calcP(deck[2]);
page 7
"The value of a hand is: "
points
points > 21
True
"You lost"
shfflDeck(deck, cards);
points = 0;
play = nxtGm();
AGAIN
False
"Would you like to draw
another card? [y/n] "
cin >> aCard;
Declare and
Initialize
Out of Loop
aCard = 'y' || aCard = 'n'
aCard != 'y' && aCard != 'n'
True
"INVALID INPUT!"
"Would you like to draw
another card? [y/n] "
cin >> aCard;
False
page 5

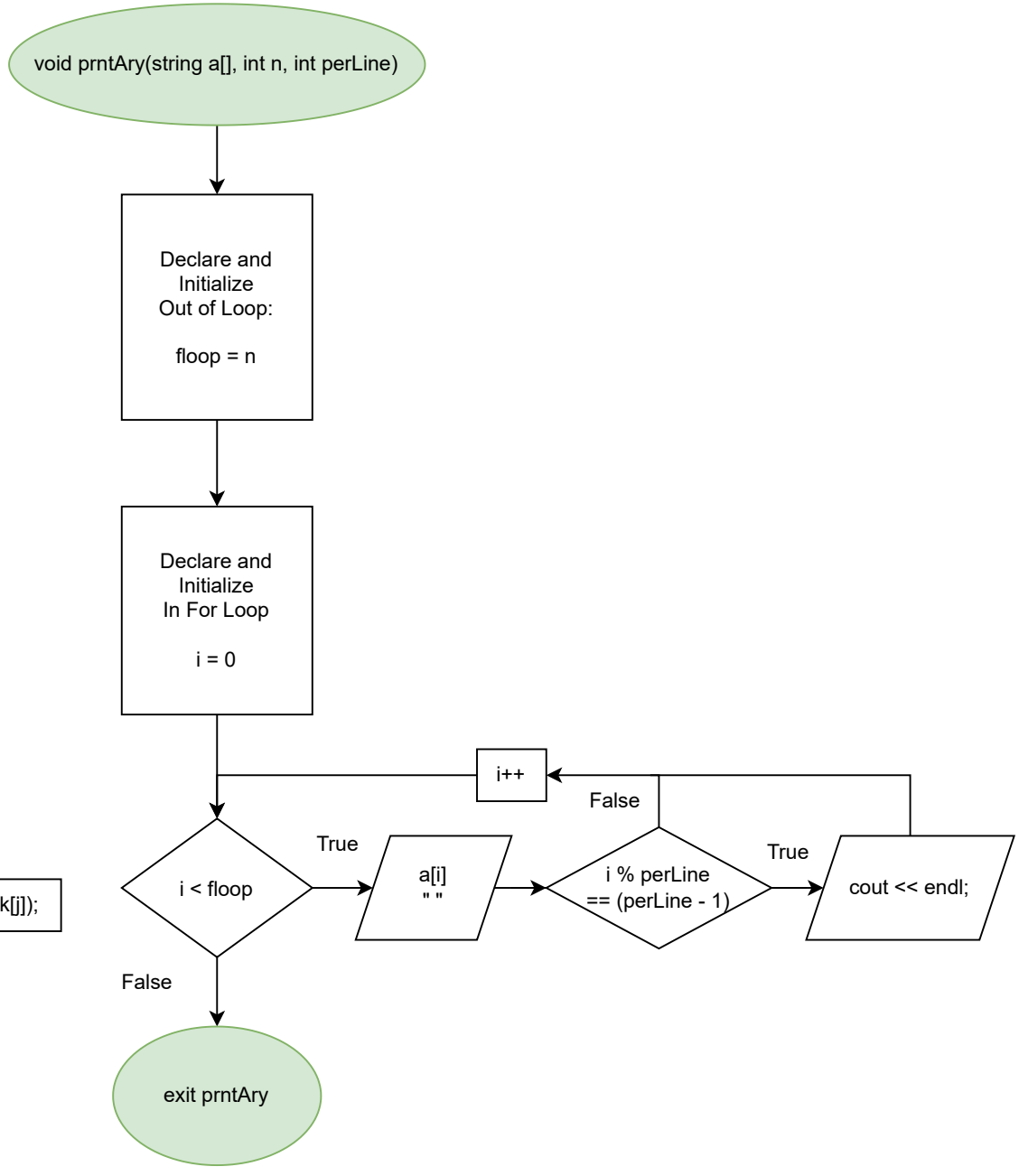

page 5
aCard == 'y'
True
cout << deck[3]
False
page 7
points += calcP(deck[3]);
"The value of a hand is: "
points
points > 21
True
"You lost"
shfflDeck(deck, cards);
points = 0;
play = nxtGm();
AGAIN
False
"Would you like to draw
another card? [y/n] "
cin >> aCard;
Declare and
Initialize
Out of Loop
aCard = 'y' || aCard = 'n'
aCard != 'y' && aCard != 'n'
True
"INVALID INPUT!"
"Would you like to draw
another card? [y/n] "
cin >> aCard;
False
page 6

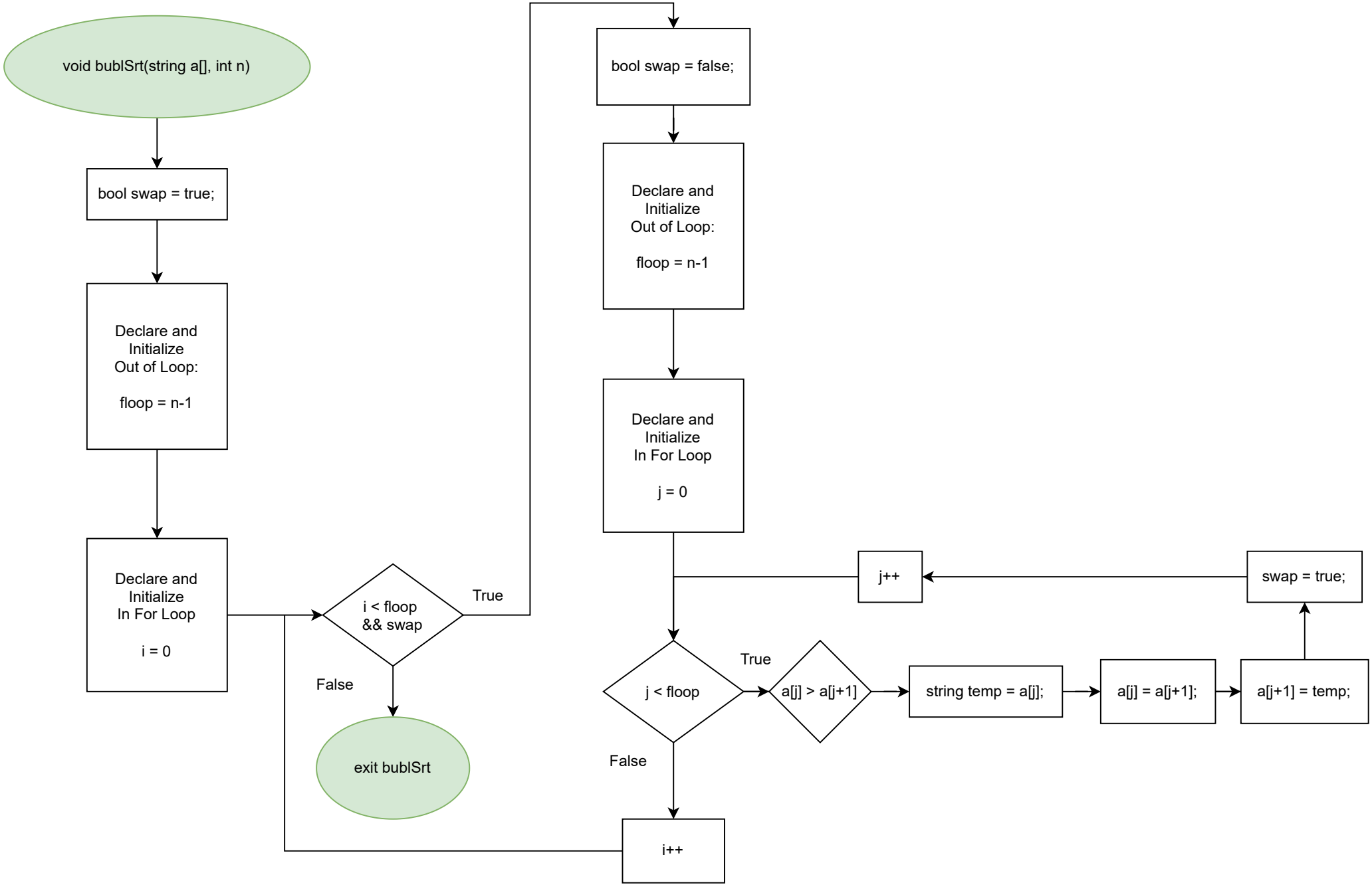

page 6
aCard == 'y'
True
cout << deck[4]
points += calcP(deck[4]);
"The value of a hand is: "
points
False
page 7

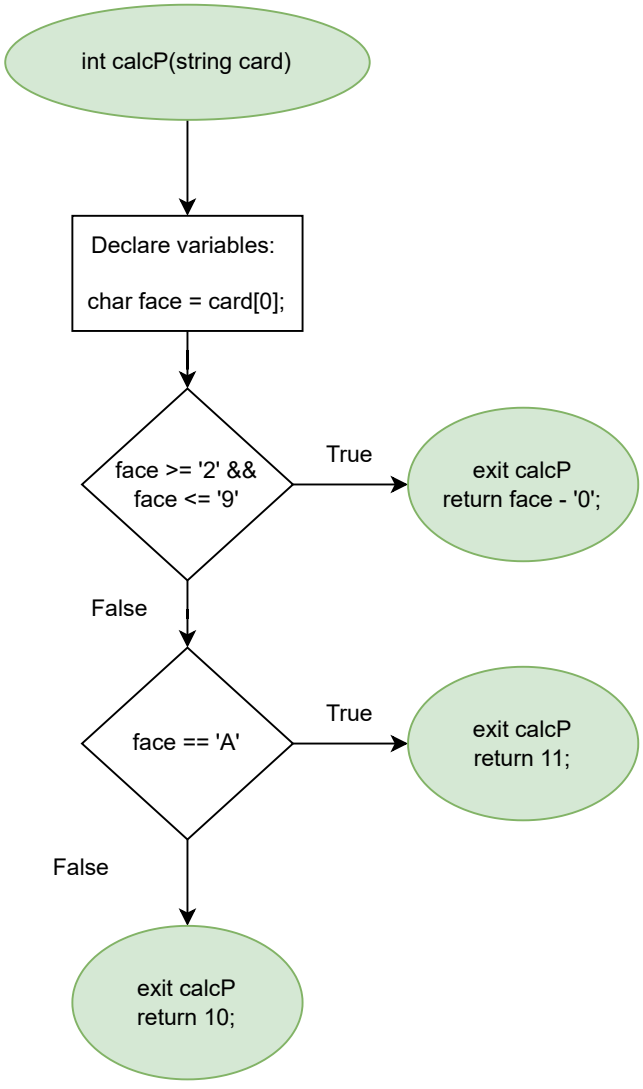

page 7
"The dealer has: "
dealer
points > dealer &&
points <= 21
True
"Congratulations! You won!"
False
points < dealer ||
points > 21
True
"You lost!"
False
"It's a draw."
shffIDeck(deck, cards);
points = 0;
play = nxtGm();
AGAIN

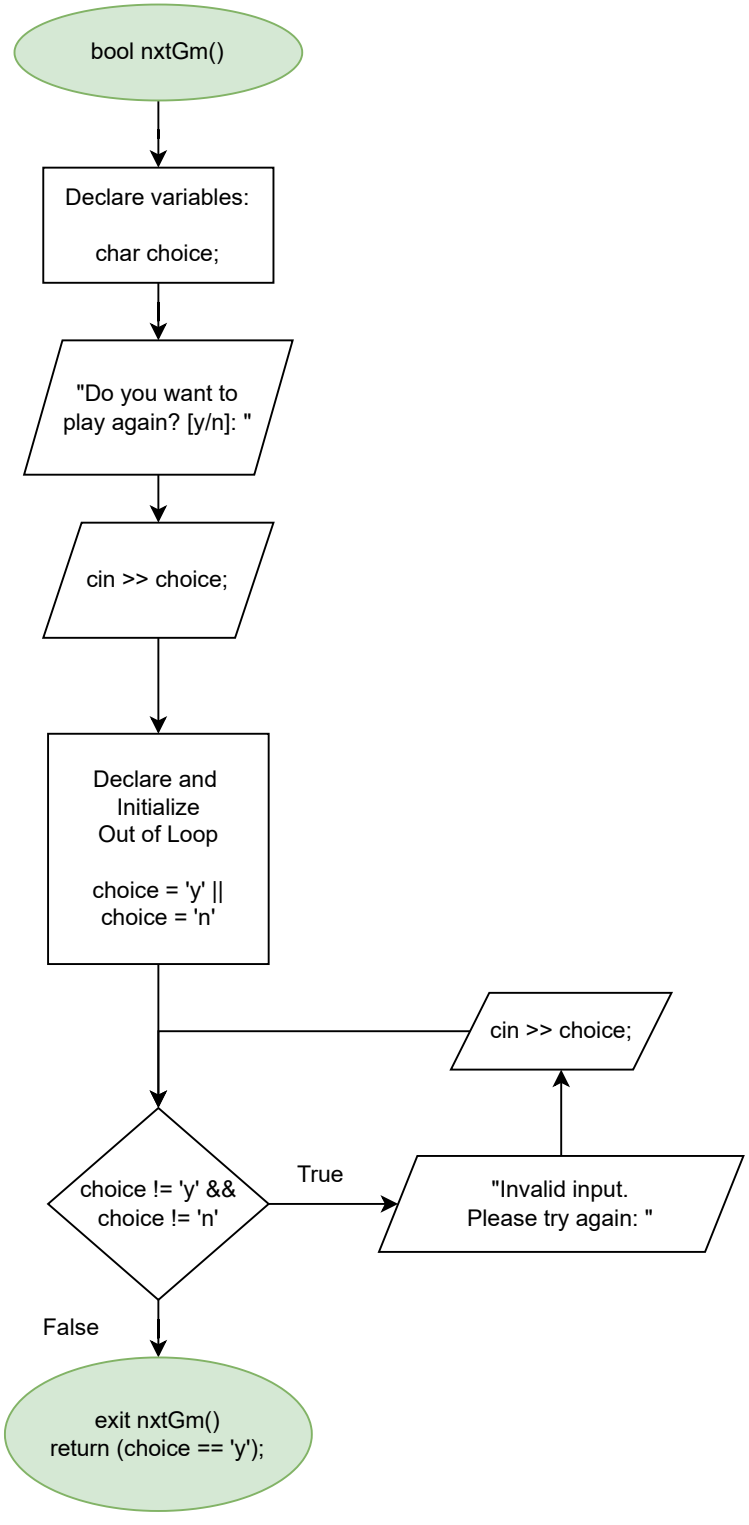

page 8
"Thanks for the game!"
Declare and Initialize Out of Loop:
floop = 50
Declare and Initialize In For Loop
i = 0
i++
i <= floop
True
" * "
False
return 0
exit from main
void genDeck(string deck[], int size)
Declare variables:
string suits[] = {"♠", "♣", "♥", "♦"};
string faces[] = {"2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K", "A"};
Declare and Initialize Out of Loop:
floop = 4
Declare and Initialize In For Loop
i = 0
i < floop
True
False
exit genDeck
Declare and Initialize Out of Loop:
floop = 13
Declare and Initialize In For Loop
j = 0
j++
j < floop
True
deck[index] = faces[j] + suits[i]; index++;
False
i++


void shfflDeck(string deck[], int size)
Declare and Initialize Out of Loop:
floop = 0
Declare and Initialize In For Loop
i = size - 1
i--
i > floop
True
int j = rand() % (i + 1);
swap(deck[i], deck[j]);
False
exit shfflDeck


void prntAry(string a[], int n, int perLine)
Declare and Initialize Out of Loop:
floop = n
Declare and Initialize In For Loop
i = 0
i++
i < floop
True
a[i] " "
i % perLine == (perLine - 1)
True
cout << endl;
False
False
exit prntAry


void bublSrt(string a[], int n)
bool swap = true;
Declare and Initialize Out of Loop:
floop = n-1
Declare and Initialize In For Loop
i = 0
i < floop && swap
True
False
exit bublSrt
bool swap = false;
Declare and Initialize Out of Loop:
floop = n-1
Declare and Initialize In For Loop
j = 0
j < floop
True
a[j] > a[j+1]
string temp = a[j];
a[j] = a[j+1];
a[j+1] = temp;
swap = true;
j++
False
i++


int calcP(string card)
Declare variables:
char face = card[0];
face >= '2' &&
face <= '9'
True
exit calcP
return face - '0';
False
face == 'A'
True
exit calcP
return 11;
False
exit calcP
return 10;


bool nxtGm()
Declare variables:
char choice;
"Do you want to
play again? [y/n]: "
cin >> choice;
Declare and
Initialize
Out of Loop
choice = 'y' ||
choice = 'n'
cin >> choice;
choice != 'y' &&
choice != 'n'
True
"Invalid input.
Please try again: "
False
exit nxtGm()
return (choice == 'y');

Cross Reference

Chapter	Section	Topic	Where Line #'s	Example
6		Functions		

	3	Function Prototypes	19 - 24	void genDeck(string [], int);
				void shfflDeck(string [], int);
				void prntAry(string [],int,int);
				void bublSrt(string [],int);
				int calcP(string card);
				bool nxtGm();

	5	Pass by Value	305 - 320	void genDeck(string deck[], int size) {
				
				string suits[] =
				{"♠", "♣", "♥", "♦"};
				string faces[] =
				{"2", "3", "4", "5", "6", "7", "8",
				"9", "10", "J", "Q", "K", "A"};
				int index = 0;
				
				for (int i = 0; i < 4; i++) {
				for (int j = 0; j < 13; j++) {
				deck[index] =
				faces[j] + suits[i];
				index++;
				}
				}
				}

	8	Return	366 - 386	int calcP(string card) {
				
				char face = card[0];
				
				if (face >= '2' && face <= '9') {
				return face - '0';
				}
				
				else if (face == 'A') {
				return 11;
				}
				
				else {
				return 10;
				}
				}

Chapter	Section	Topic	Where Line #'s	Example

	9	Returning boolean	389	bool nxtGm() {
				
				char choice;
				cout << "Do you want to play again? [y/n]: ";
				cin >> choice;
				
				while (choice != 'y' && choice != 'n') {
				cout << "Invalid input. Please try again: ";
				cin >> choice;
				}
				
				return (choice == 'y');
				}

	11	Static variables	310	static int index = 0;
	15	Exit() function	96	if (start != 'y') exit(0);
7		Arrays		
	1 to 6	1D Array	308	string suits[] = {"♠", "♣", "♥", "♦"};

	7	Parallel Arrays	305 - 320	void genDeck(string deck[], int size) {
				
				string suits[] = {"♠", "♣", "♥", "♦"};
				string faces[] = {"2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K", "A"};
				int index = 0;
				
				for (int i = 0; i < 4; i++) {
				for (int j = 0; j < 13; j++) {
				deck[index] = faces[j] + suits[i];
				index++;
				}
				}
				}

Chapter	Section	Topic	Where Line #'s	Example

	8	1D as Function Arguments	332 - 340	void prntAry(string a[], int n, int perLine) {
				
				for (int i = 0; i < n; i++) {
				cout << setw(5) << a[i] << " ";
				if (i % perLine == (perLine - 1)) cout << endl;
				}
				}

	9	2D Arrays	73 - 79	string arr[7][2] = { "a) ", "You will be given two cards. ", "b) ", "You will see the value of the hand. ", "c) ", "The program will ask you if you want to draw more cards.", "d) ", "Type y for YES and n for NO. ", "e) ", "You can draw only three more cards from the deck. ", "f) ", "The program will display the results. ", "g) ", "If you want to play again, type y for YES and n for NO. "} ;

		Passing Arrays to and from Functions	323 - 329	void shfflDeck(string deck[], int size) {
				
				for (int i = size - 1; i > 0; i--) {
				int j = rand() % (i + 1);
				swap(deck[i], deck[j]);
				}
				}

8		Searching and Sorting Arrays		

Chapter	Section	Topic	Where Line #'s	Example

	3	Bubble Sort	343 - 363	void bublSrt(string a[], int n) {
				
				bool swap = true;
				for(int i = 0; i < n-1 && swap; i++) {
				bool swap = false;
				
				for (int j = 0; j < n-1; j++) {
				if (a[j] > a[j+1]) {
				
				string temp = a[j];
				a[j] = a[j+1];
				a[j+1] = temp;
				
				swap = true;
				}
				}
				}
				}

	3	Selection Sort	323 - 329	void shfflDeck(string deck[], int size) {
				
				for (int i = size - 1; i > 0; i--) {
				int j = rand() % (i + 1);
				swap(deck[i], deck[j]);
				}
				}

Pseudo Code

```
/*
 * File:   main.cpp
 * Author: Diana Marciniak
 * Created on Feb 10, 11:48 AM
 * Purpose : Project2 Final Version
 */

// System Libraries
// I/O Library
// Random Function Library
// Time Library
// Math Library
```

```
// Formatting Library


// Global Constants - Math Physics, Chemistry, Conversions

// Function Prototypes Begin Here

// Program Execution Begins Here

        // Set a random seed

        // Declare all variables
                // Number of cards
                // The user's points start from 0
                // The dealer's points
                // Number of cards
                // Start the game [y/n]
                // Play again [y/n]
                // Another card [y/n]
                // Probability of winning the game is 42.22%
                // Next Game
                // Size of the deck
                / The deck of cards

        // Generate the deck of cards

        // Bubble Sort

        //Print the array

        // Shuffle the deck

        // Print out "*"

        // Play the game

        // The introduction

        // Asking the user if they want to start the game

        // Dealer's random number of points

        // Two random cards

        // Points for the first two cards

        // If points = 21 - Blackjack; the user wins

        // Else, continue the game

        // The value of a hand is the sum of the points of deck[1] and deck[2]

        // If the input is invalid, repeat the question
```

```
        // Drawing a third card - yes
                // If points are > 21 - end the game
                // If the points are not > 21 - continue the game
                // User chooses if he/she wants to draw another card
                // If the input is invalid, repeat the question

        // Drawing a fourth card - yes
                // The user's points
                // If points are > 21 - end the game
                // If the points are not > 21 - continue the game
                // User chooses if he/she wants to draw another card
                // If the input is invalid, repeat the question
        // Drawing a fifth card - yes
                // The user's points
                // The dealer's points
                // If the user has more points than the dealer and the points are <= 21 - the user
                   wins
                //If the user has less points than the dealer or the points are > 21 - the user
                  looses
                // If user's points = the dealer's points

        // Drawing a fifth card - no
                // The dealer's points
                // If the user has more points than the dealer and the points are <= 21 - the user
                   wins
                //If the user has less points than the dealer or the points are > 21 - the user
                  looses
                // If user's points = the dealer's points

        // Drawing a fourth card - no
                // The dealer's points
                // If the user has more points than the dealer and the points are <= 21 - the user
                  wins
                //If the user has less points than the dealer or the points are > 21 - the user
                  looses
                // If user's points = the dealer's points

        // Drawing a third card - no
                // The dealer's points
                // If the user has more points than the dealer and the points are <= 21 - the user
                   wins
                //If the user has less points than the dealer or the points are > 21 - the user
                  looses
                // If user's points = the dealer's points


        // Shuffle the deck again

        // Set points equal to 0 for the next game

        // End the game

//Exit the Program
```

```
// Function to generate a deck of cards

        // Declare variables
// Exit the program

// Function to shuffle the deck
// Exit the program

// Function to print the array
// Exit the program

// Function with Bubble Sort

        // Declare variables
// Exit the program

// Function calculating points

        // Declare variables
        // Calculation
        //Exit1
        //Exit2
        //Exit3

// Function - another game [y/n]

        // Declare variables
        // Check if the input is valid
// Exit the program
```

Reference

- Github

- https://www.blackjackinfo.com/blackjack-rules/