Blackjack Card Game

Jordan Moreno

CIS-17A-45398

Summer 2021

Introduction: Blackjack is a card playing game where the goal is to have your card points as close to 21 as possible to beat the dealer. Your total points are gathered by adding up your card values, with Aces being worth either 1 or 11 points, number cards being worth their regular number value, and Face cards all being worth 10 points.

You start with 2 cards and the dealer starts with 1, from there you can decide to either ‘hit’ for another card, or ‘stand’ to continue with the cards you already have. If your points are above 21 after hitting, you’ve busted, and you lose your wager. Once you choose to ‘stand’, the dealer's cards are revealed, and if your points are higher than that of the dealers and under 21, or if the dealer's points are above 21 and they bust, you win. The amounts you win can be determined both by your wager and whether or not you get a Blackjack (a Blackjack is 21 points).

My Approach: I wanted to make a player vs. computer game for blackjack, so I started by creating a function that creates a new, non-randomized shoe of cards, times four (the regular amount of shoes used in a Blackjack game) and then creating a function that will shuffle them. Afterwards, two random cards are assigned to the Player, and one to the Dealer. As the Player chooses to hit, a new card is given, until they bust and lose their wager, win blackjack, or choose to stand, where then the Dealer’s second card will be revealed and have the count compared to the Player’s count to determine who has won.

I added an enumeration for the Face cards to keep their numbers simple and a structure array that stores a user input from a binary file and text file for the Player’s name and starting balance.

Balance start: The player’s starting balance can be changed via input from a binary file and is defaulted at $500;

Winning a game results in the player’s wager being returned and the same amount given back, thus winning x2 what they put in.

Losing a game results in the wager being lost.

Winning a game with a blackjack, or 21 points, results in the original wager being returned plus 1.5x amount given back, thus winning 2.5x total of what was put in.

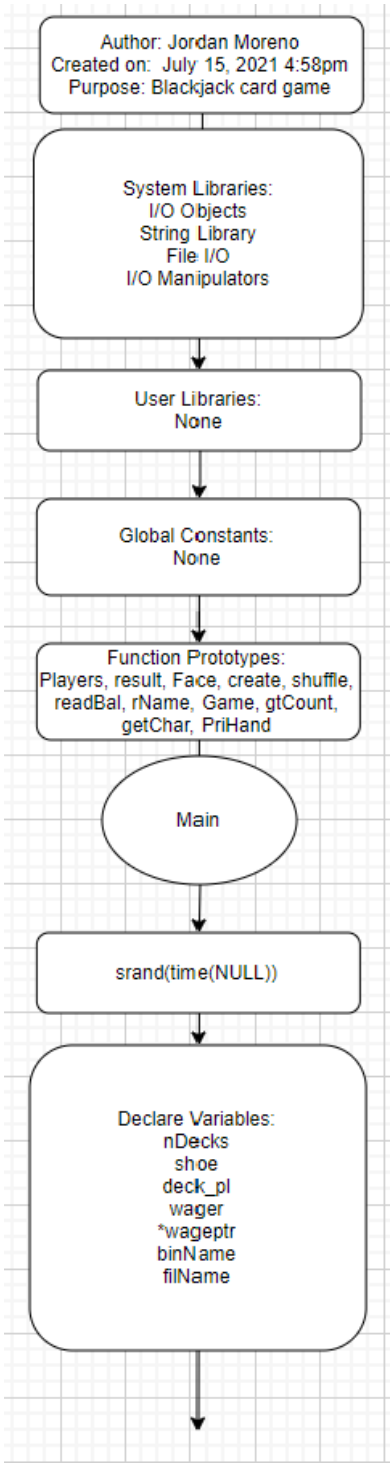
Cross Reference for Project 2

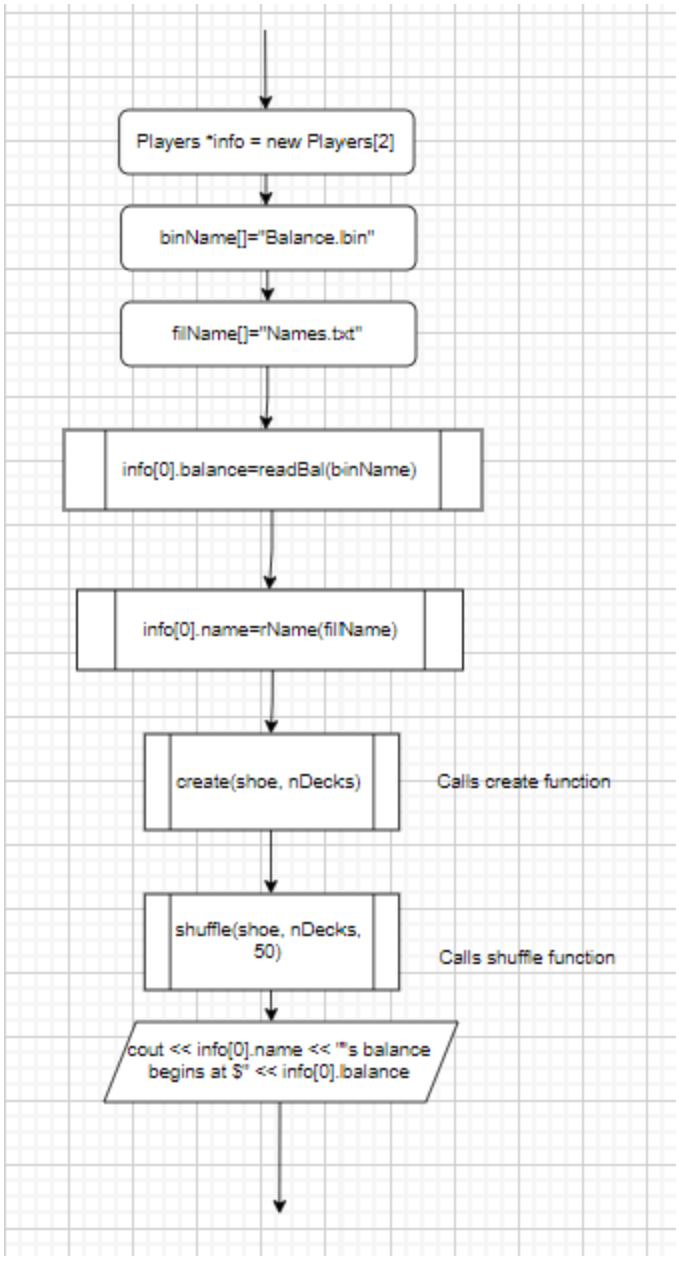
You are to fill-in with where located in code

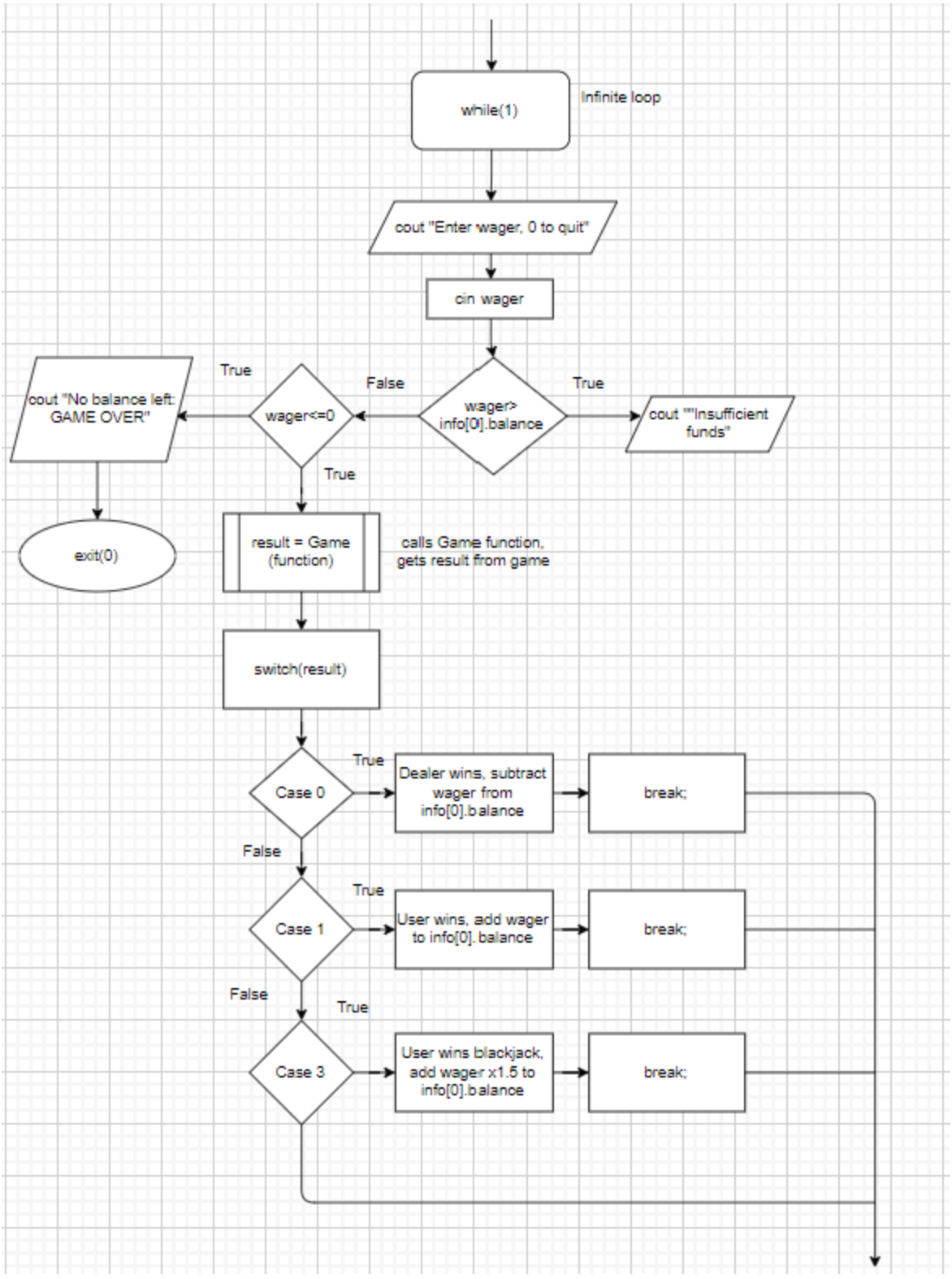
Chapter	Section	Topic	Where Line #"s	Pts	Notes
13		Classes			
	1 to 3	Instance of a Class	25,38,93	4	
	4	Private Data Members	40,41,95,96	4	Never Public
	5	Specification vs. Implementation	.h file/25,38,93	4	.h vs. .cpp files Always split
	6	Inline	29,32,51,105	4	
	7, 8, 10	Constructors	29, 43,98,99	4	Overloading
	9	Destructors	50	4	
	12	Arrays of Objects	288	4	
	16	UML		4	
14		More about Classes			
	1	Static		5	
	2	Friends	51,105	2	
	4	Copy Constructors		5	
	5	Operator Overloading	49,51,105	8	Overload 3 operators
	7	Aggregation		6	

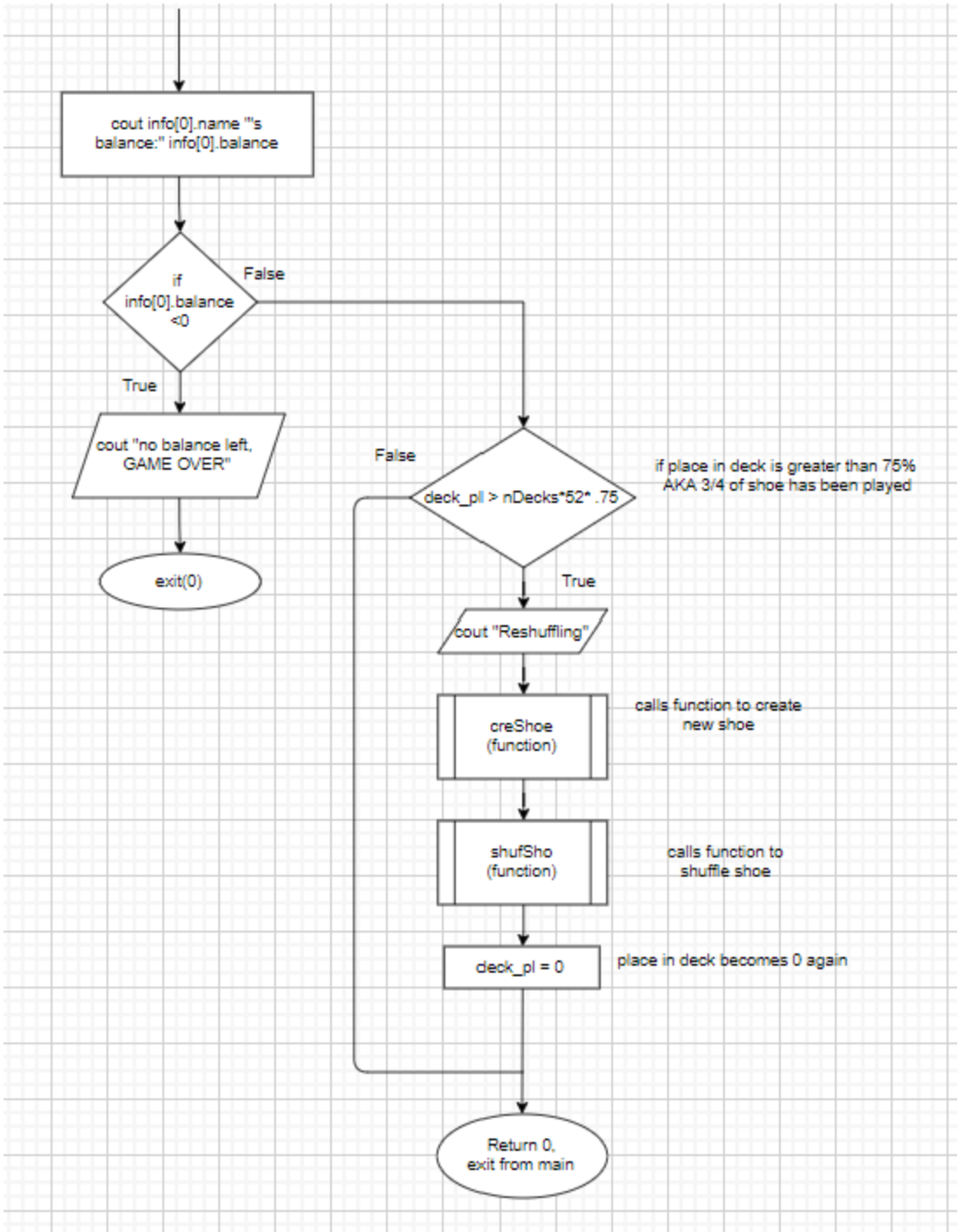
15		Inheritance			
	1	Protected members	27	6	
	2 to 5	Base Class to Derived	25,38	6	
	6	Polymorphic associations		6	
	7	Abstract Classes		6	
16		Advanced Classes			
	1	Exceptions	118	6	
	2 to 4	Templates	161	6	
	5	STL		6	
		Sum		100	

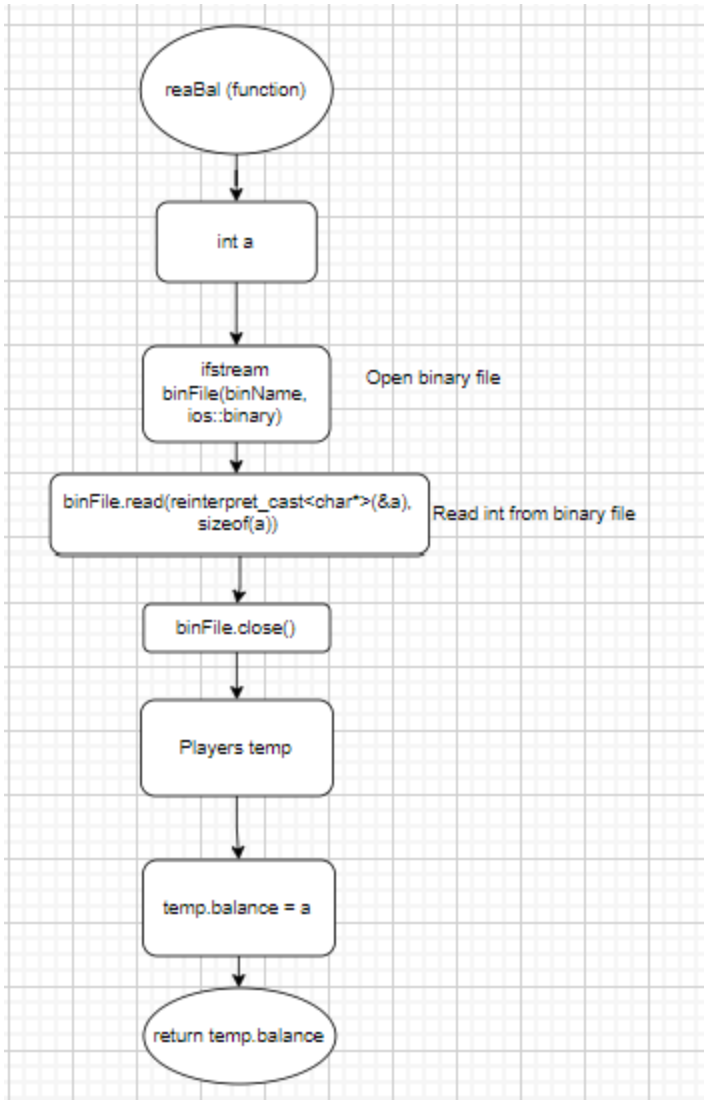
Flowchart:

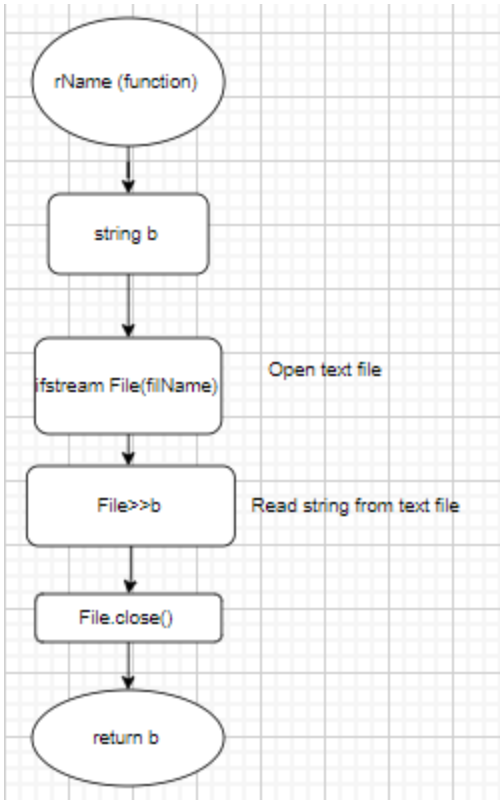

Author: Jordan Moreno
Created on: July 15, 2021 4:58pm
Purpose: Blackjack card game
System Libraries:
I/O Objects
String Library
File I/O
I/O Manipulators
User Libraries:
None
Global Constants:
None
Function Prototypes:
Players, result, Face, create, shuffle, readBal, rName, Game, gtCount, getChar, PriHand
Main
srand(time(NULL))
Declare Variables:
nDecks
shoe
deck_pl
wager
*wageptr
binName
filName

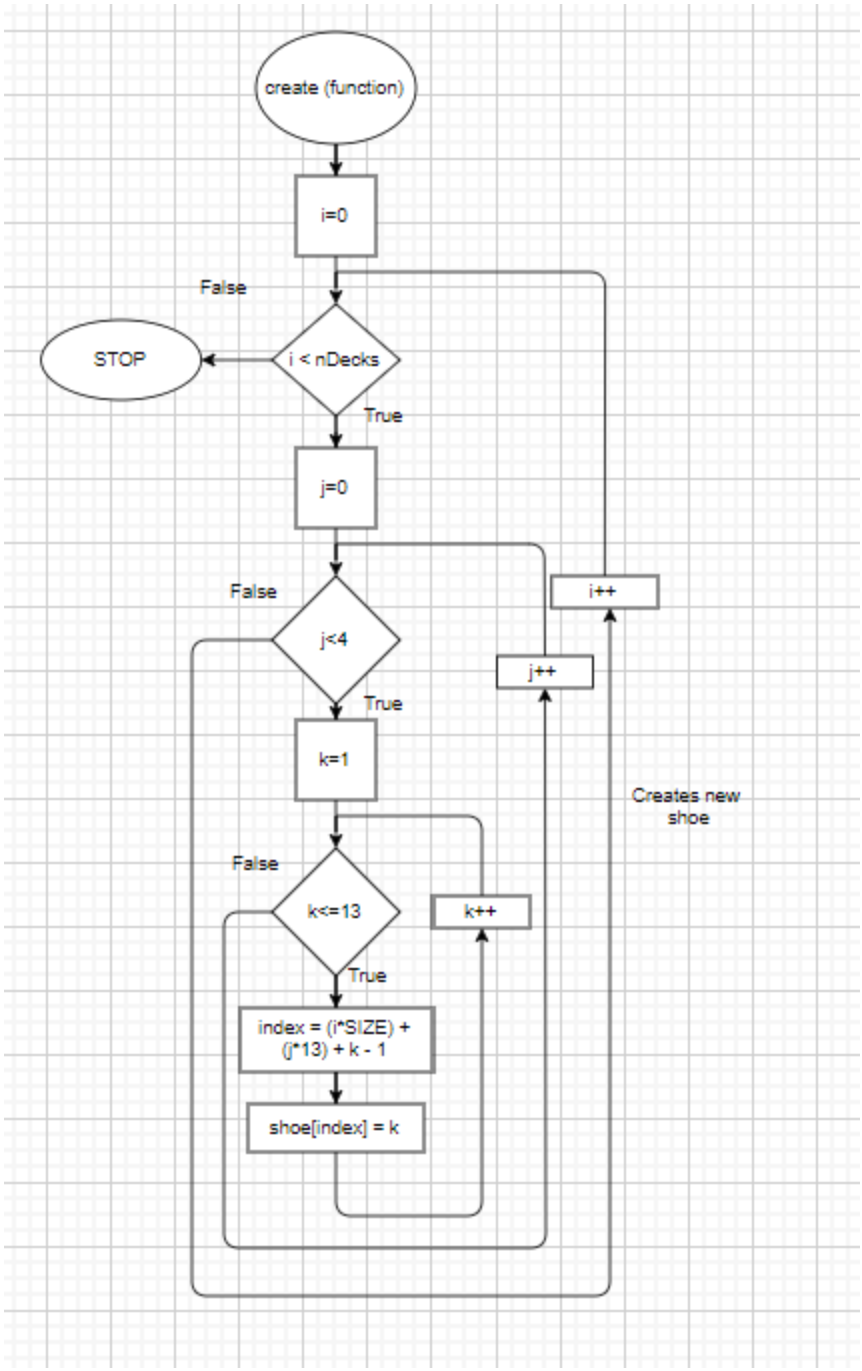

Players *info = new Players[2]
binName[]="Balance.bin"
filName[]="Names.txt"
info[0].balance=readBal(binName)
info[0].name=rName(filName)
create(shoe, nDecks)
Calls create function
shuffle(shoe, nDecks, 50)
Calls shuffle function
cout << info[0].name << "'s balance begins at $" << info[0].balance

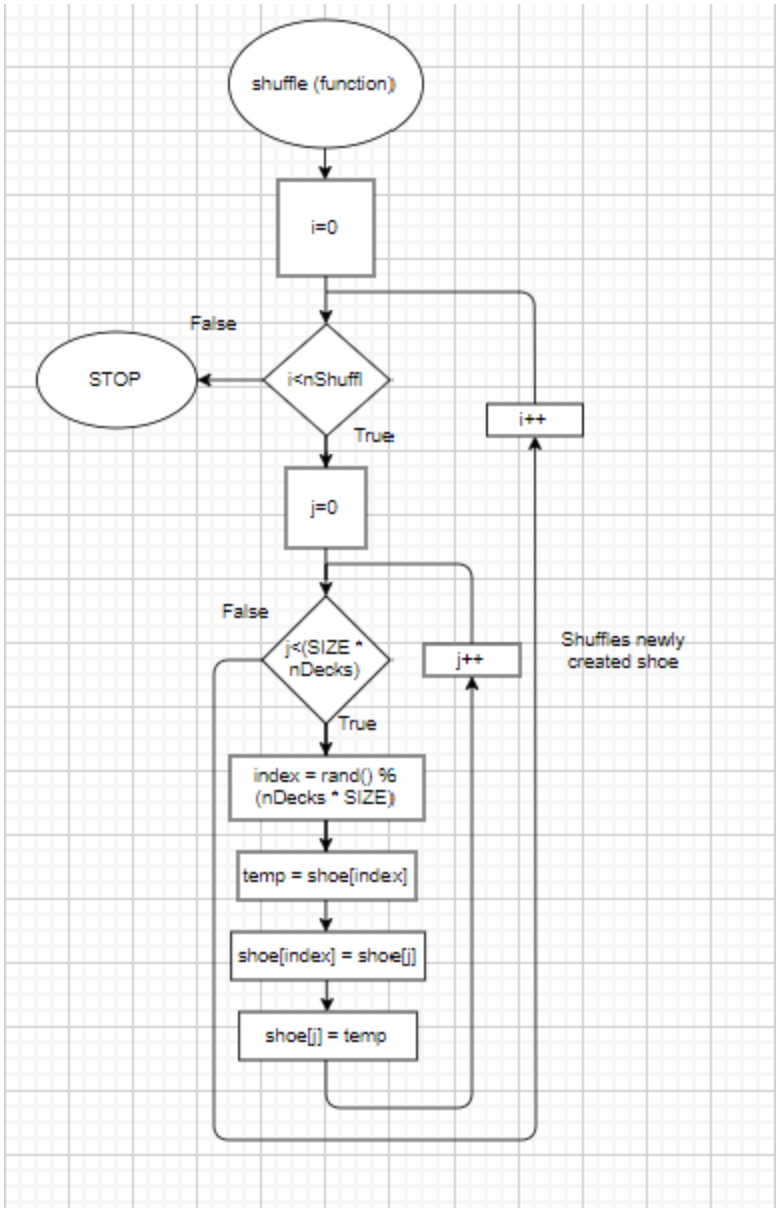

while(1)
Infinite loop
cout "Enter wager, 0 to quit"
cin wager
wager> info[0].balance
True
cout ""Insufficient funds"
False
wager<=0
True
cout "No balance left: GAME OVER"
exit(0)
True
result = Game (function)
calls Game function, gets result from game
switch(result)
Case 0
True
Dealer wins, subtract wager from info[0].balance
break;
False
Case 1
True
User wins, add wager to info[0].balance
break;
False
Case 3
True
User wins blackjack, add wager x1.5 to info[0].balance
break;

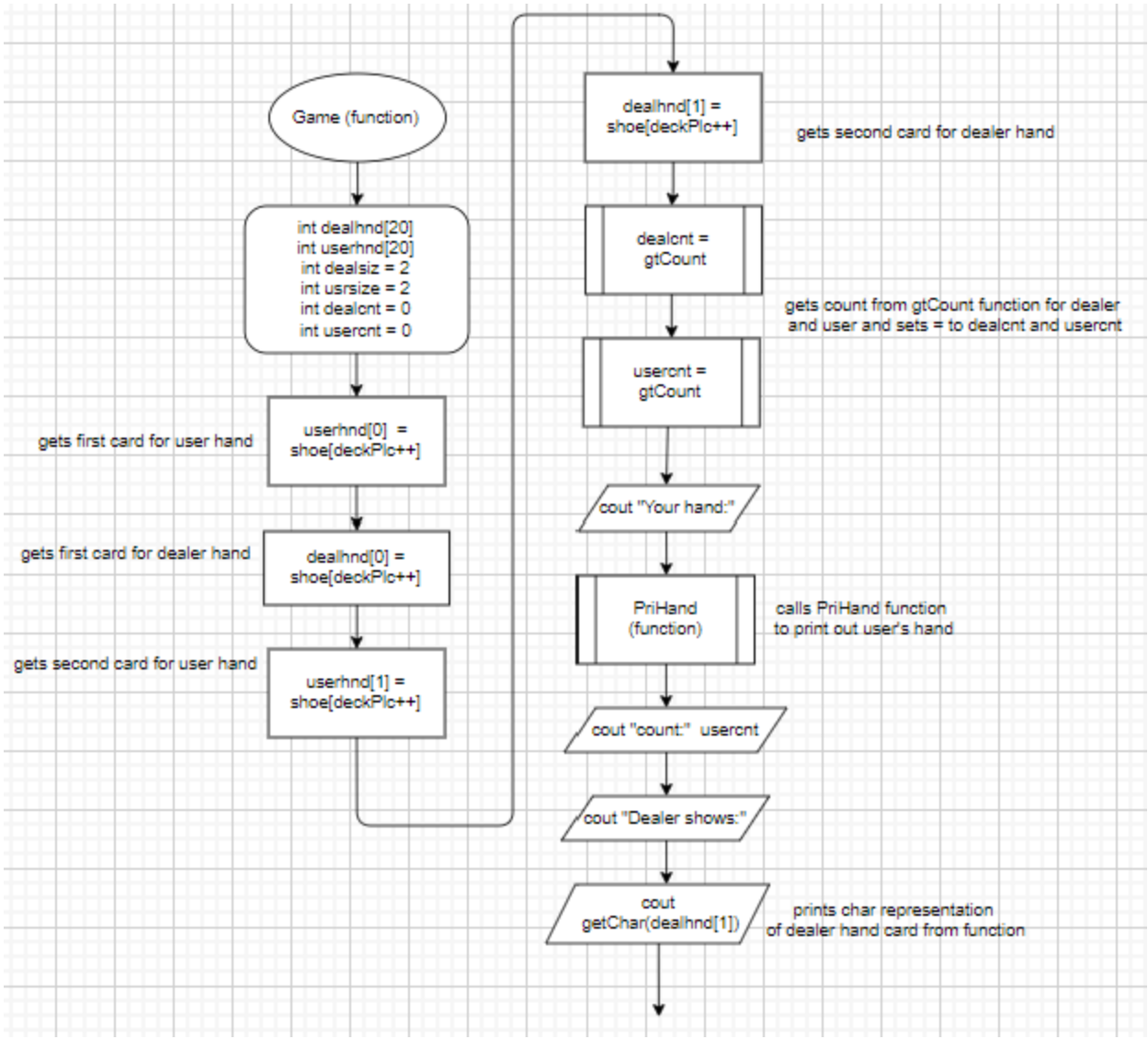

cout info[0].name "'s balance:" info[0].balance
if info[0].balance <0
False
True
cout "no balance left, GAME OVER"
exit(0)
False
deck_pl > nDecks*52* .75
if place in deck is greater than 75% AKA 3/4 of shoe has been played
True
cout "Reshuffling"
creShoe (function)
calls function to create new shoe
shufSho (function)
calls function to shuffle shoe
deck_pl = 0
place in deck becomes 0 again
Return 0, exit from main

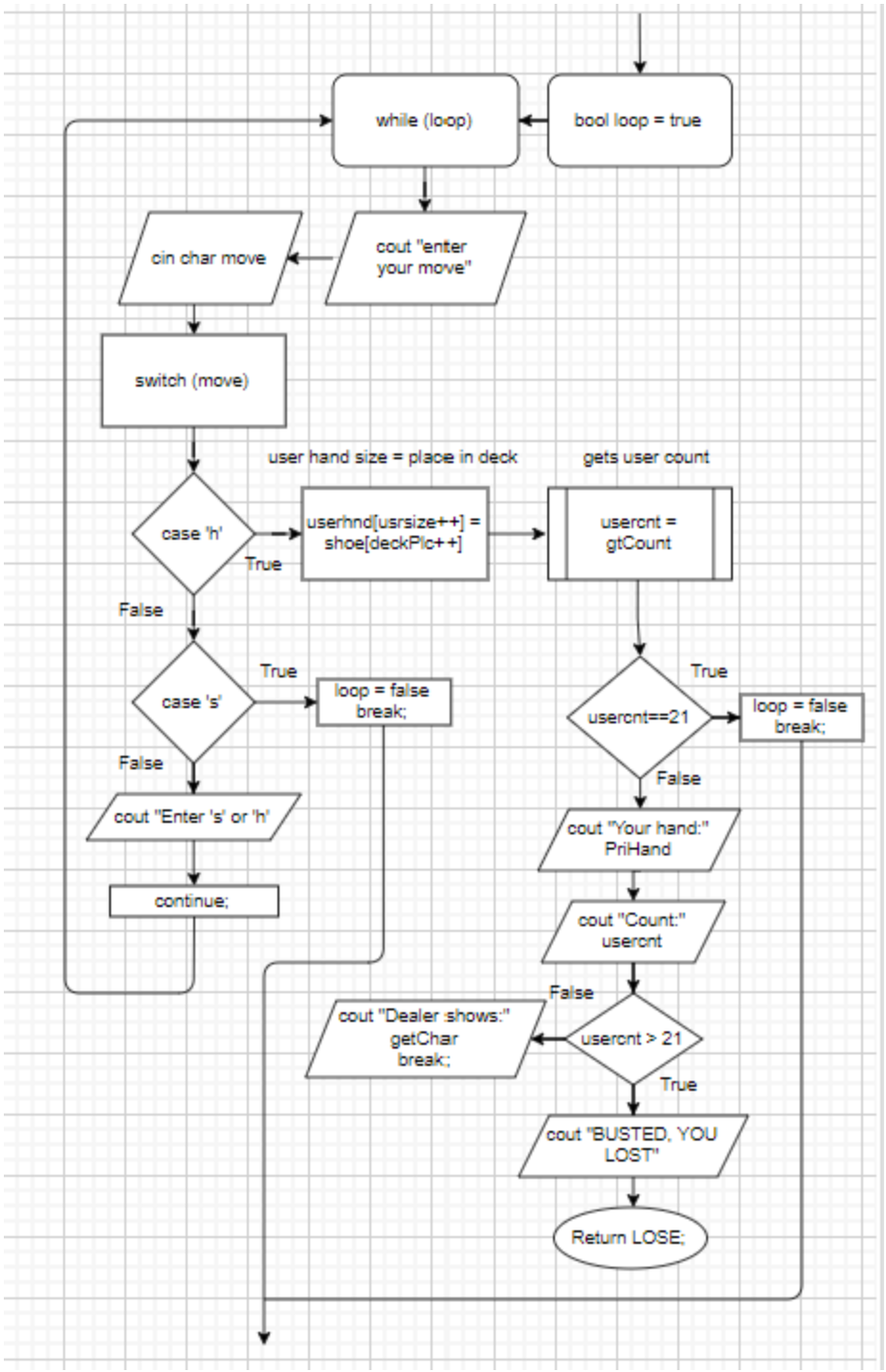

reaBal (function)
int a
ifstream binFile(binName, ios::binary)
Open binary file
binFile.read(reinterpret_cast<char*>(&a), sizeof(a))
Read int from binary file
binFile.close()
Players temp
temp.balance = a
return temp.balance

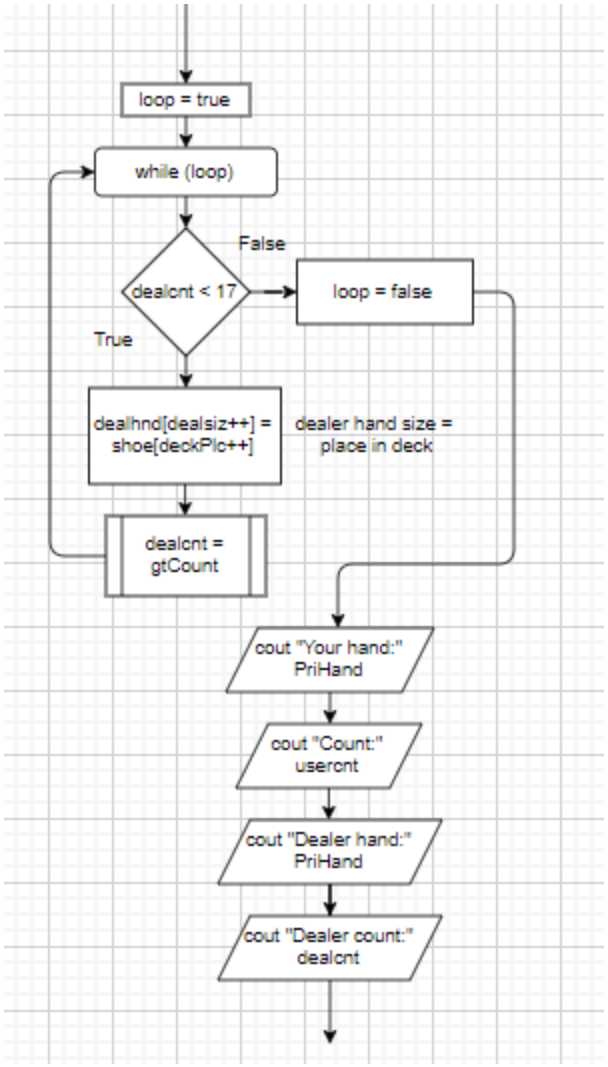

rName (function)
string b
ifstream File(filName)
Open text file
File>>b
Read string from text file
File.close()
return b

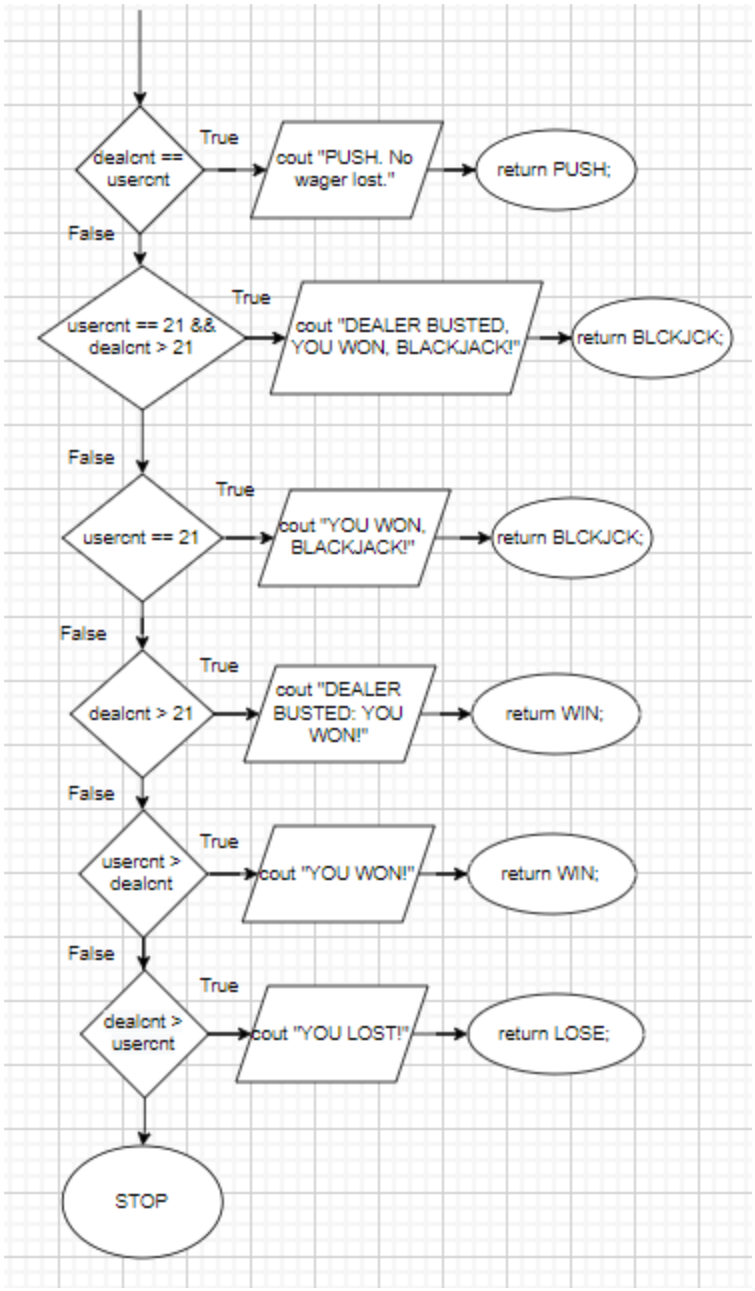

create (function)
i=0
False
STOP
i < nDecks
True
j=0
False
j<4
True
k=1
False
k<=13
True
index = (i*SIZE) +
(j*13) + k - 1
shoe[index] = k
k++
j++
i++
Creates new
shoe

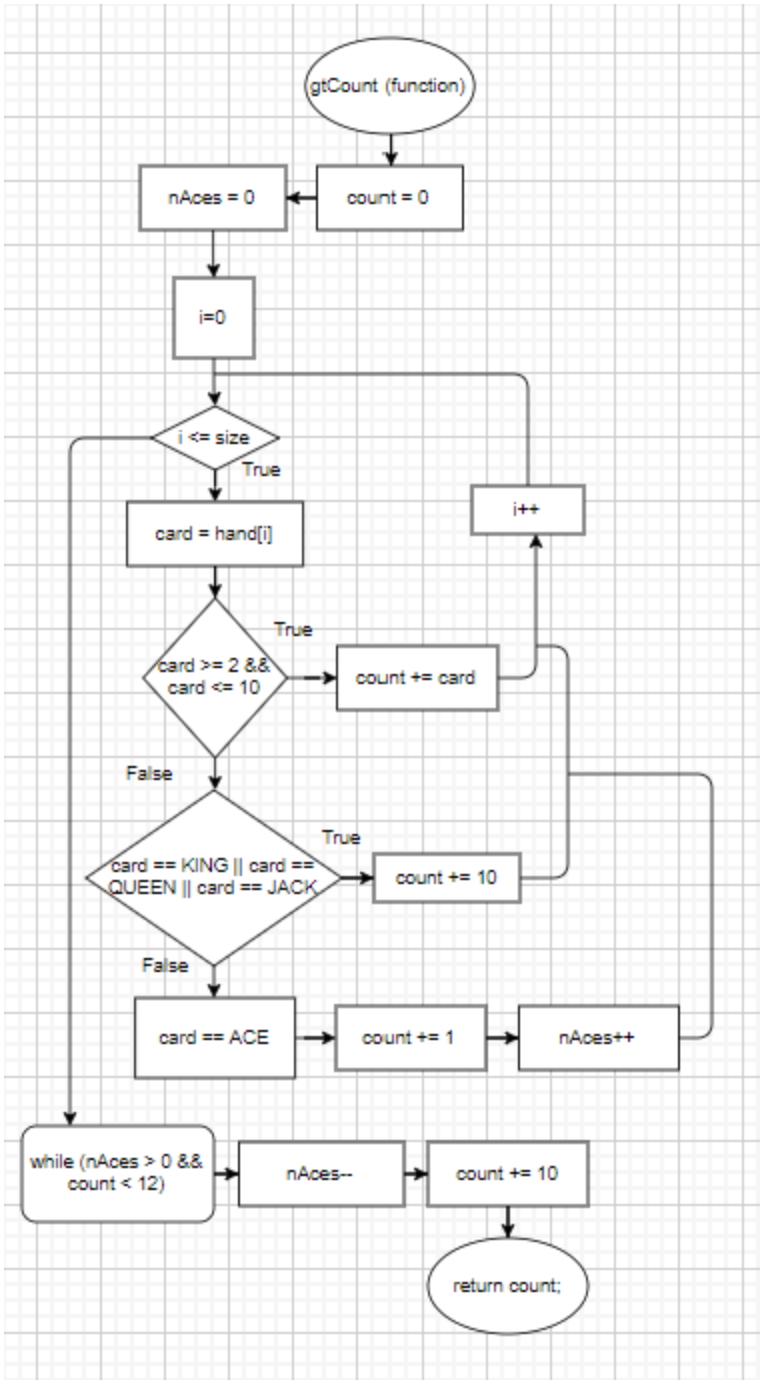

shuffle (function)
i=0
False
STOP
i<nShuffl
True
i++
j=0
False
j<(SIZE * nDecks)
j++
Shuffles newly created shoe
True
index = rand() % (nDecks * SIZE)
temp = shoe[index]
shoe[index] = shoe[j]
shoe[j] = temp

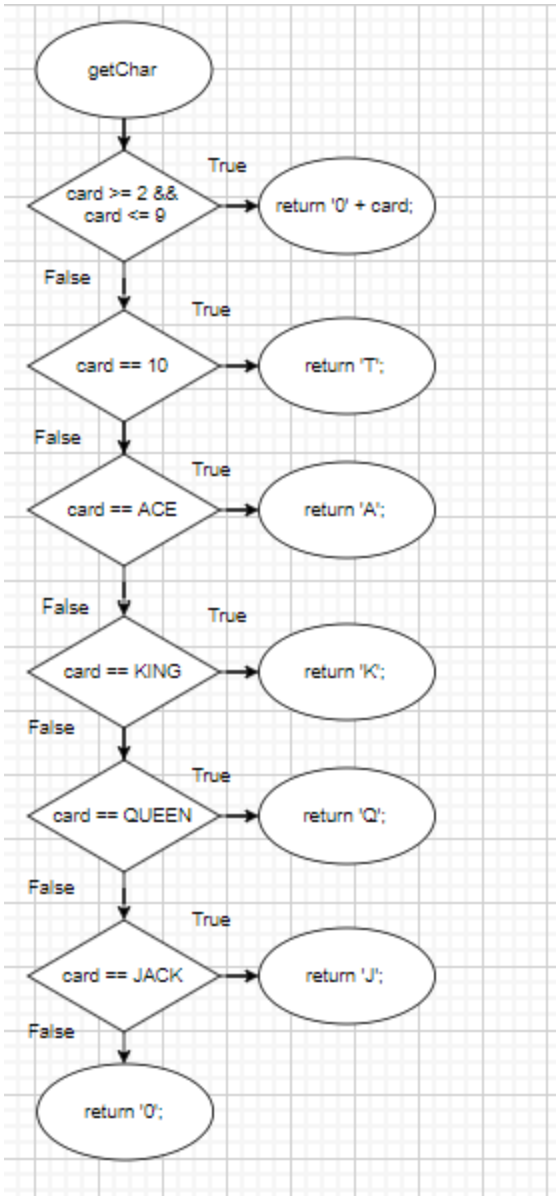

Game (function)
int dealhnd[20]
int userhnd[20]
int dealsiz = 2
int usrsize = 2
int dealcnt = 0
int usercnt = 0
gets first card for user hand
userhnd[0] = shoe[deckPlc++]
gets first card for dealer hand
dealhnd[0] = shoe[deckPlc++]
gets second card for user hand
userhnd[1] = shoe[deckPlc++]
dealhnd[1] = shoe[deckPlc++]
gets second card for dealer hand
dealcnt = gtCount
gets count from gtCount function for dealer and user and sets = to dealcnt and usercnt
usercnt = gtCount
cout "Your hand:"
PriHand (function)
calls PriHand function to print out user's hand
cout "count:" usercnt
cout "Dealer shows:"
cout getChar(dealhnd[1])
prints char representation of dealer hand card from function

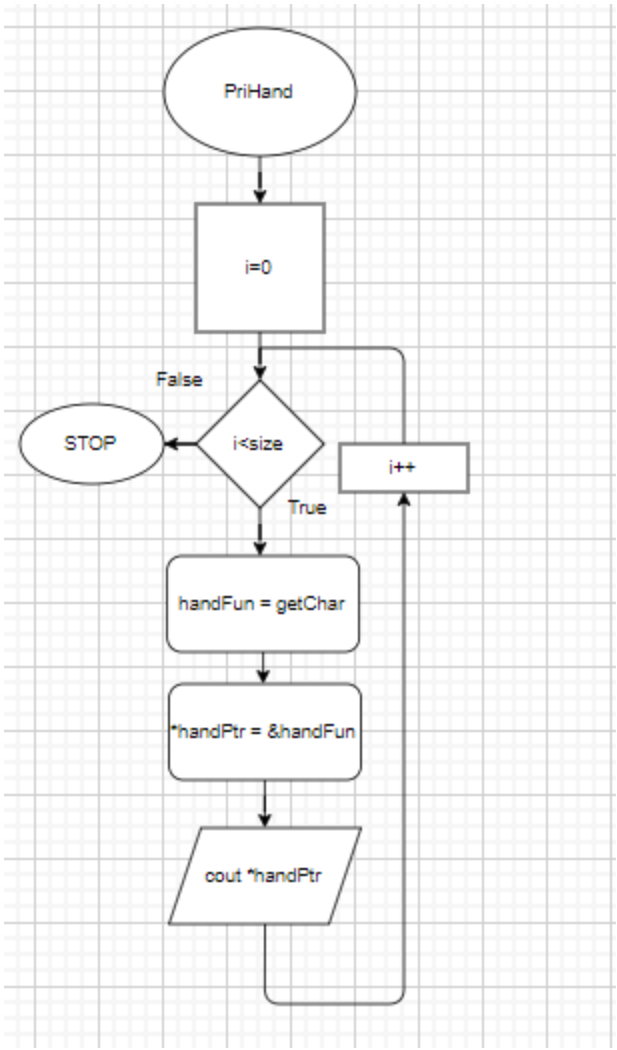

bool loop = true
while (loop)
cout "enter your move"
cin char move
switch (move)
case 'h'
True
False
user hand size = place in deck
userhnd[usrsize++] = shoe[deckPlc++]
gets user count
usercnt = gtCount
case 's'
True
False
loop = false break;
cout "Enter 's' or 'h'
continue;
usercnt==21
True
False
loop = false break;
cout "Your hand:" PriHand
cout "Count:" usercnt
usercnt > 21
False
True
cout "Dealer shows:" getChar break;
cout "BUSTED, YOU LOST"
Return LOSE;


loop = true
while (loop)
False
dealcnt < 17
loop = false
True
dealhnd[dealsiz++] = shoe[deckPlc++]
dealer hand size = place in deck
dealcnt = gtCount
cout "Your hand:" PriHand
cout "Count:" usercnt
cout "Dealer hand:" PriHand
cout "Dealer count:" dealcnt


dealcnt == usercnt
True
cout "PUSH. No wager lost."
return PUSH;
False
usercnt == 21 && dealcnt > 21
True
cout "DEALER BUSTED, YOU WON, BLACKJACK!"
return BLCKJCK;
False
usercnt == 21
True
cout "YOU WON, BLACKJACK!"
return BLCKJCK;
False
dealcnt > 21
True
cout "DEALER BUSTED; YOU WON!"
return WIN;
False
usercnt > dealcnt
True
cout "YOU WON!"
return WIN;
False
dealcnt > usercnt
True
cout "YOU LOST!"
return LOSE;
STOP


gtCount (function)
count = 0
nAces = 0
i=0
i <= size
True
card = hand[i]
i++
True
card >= 2 && card <= 10
count += card
False
True
card == KING || card == QUEEN || card == JACK
count += 10
False
card == ACE
count += 1
nAces++
while (nAces > 0 && count < 12)
nAces--
count += 10
return count;


getChar
card >= 2 && card <= 9
True
return '0' + card;
False
card == 10
True
return 'T';
False
card == ACE
True
return 'A';
False
card == KING
True
return 'K';
False
card == QUEEN
True
return 'Q';
False
card == JACK
True
return 'J';
False
return '0';


PriHand
i=0
False
STOP
i<size
i++
True
handFun = getChar
*handPtr = &handFun
cout *handPtr