Project 2

<Concentration Memory Game>

CIS5/CSC5 40404

Name: Wuysang, Emma

Date: 2/11/2024

Table of Contents

Introduction

Title: Concentration Card Game

This is a memory-based/guessing game.

The goal is to find as many pairs as possible, with the least moves possible.

With this project implementation, it will be catered to finding one of the pairs, out of all twenty pairs. A half a stack of cards, only the suit of Hearts and Spades will be shuffled (excluding Joker and Royals). 20 Cards (Ace-9) will be laid out individually.

A matching pair is defined as two cards with the same rank. (two Aces, two fives, etc.) Since there isn't a visual stack of cards, the user must type a placement (Exa. a b, c f), after choosing, the cards will be revealed.

Summary

Project size: about 573 lines

The number of variables: about 90~

Includes the 7 constructs learned.

The project took way more lines than it should have, and since we had a limited number of constructs(etc.) that we learned, I tried to utilize what I knew. I have always been enamored by memory-based games, and I remember growing up, I would play this game with a group of friends.

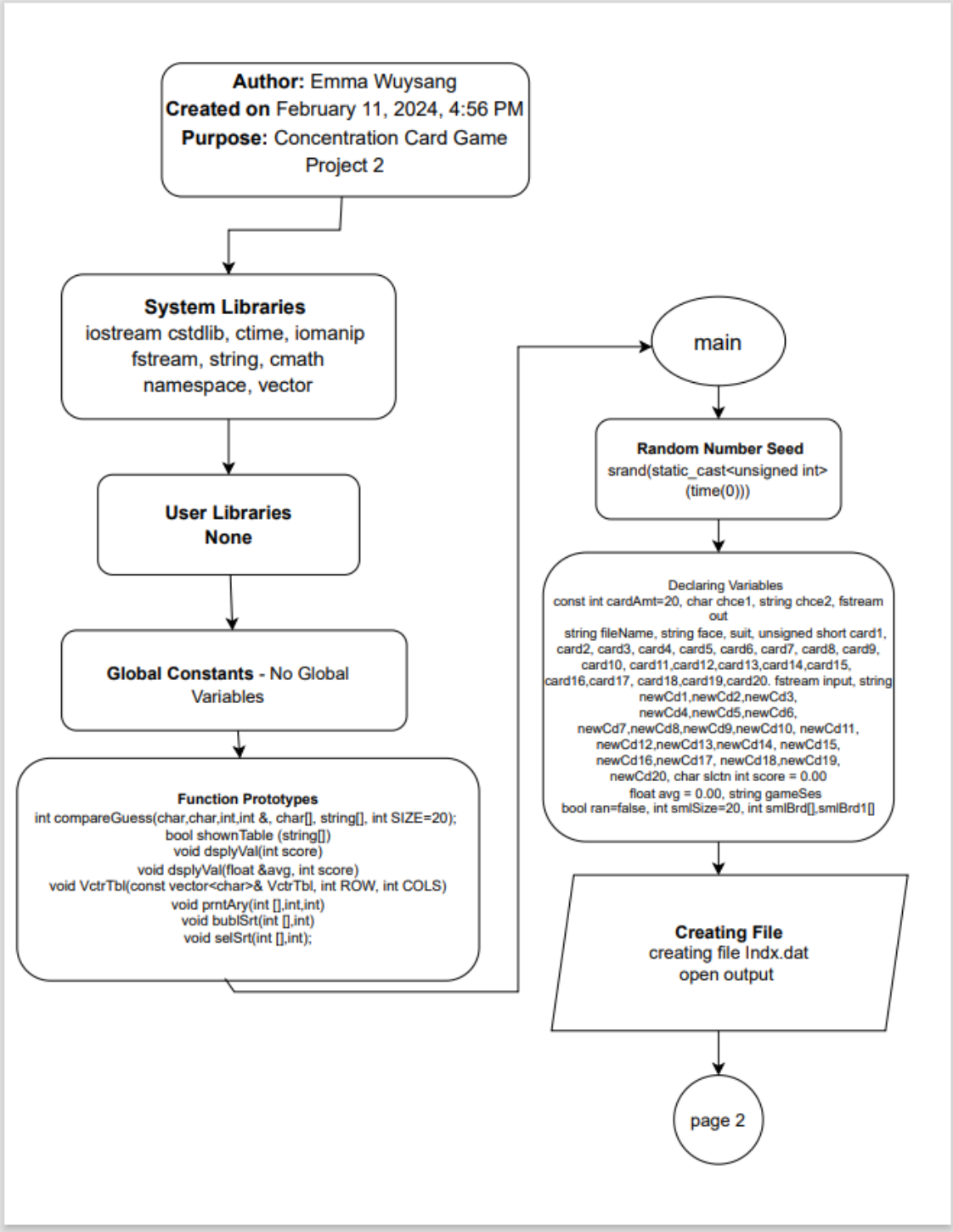
It took around a few days, and mainly hours within those days. I would have had more versions if I did not spend long periods of time working on the project. Continuously around 3-9 hours a day for around four days, I spent reviewing old material and trying to implement what I knew. I had no previous C++ experience, which made it a bit difficult to take in as much information.

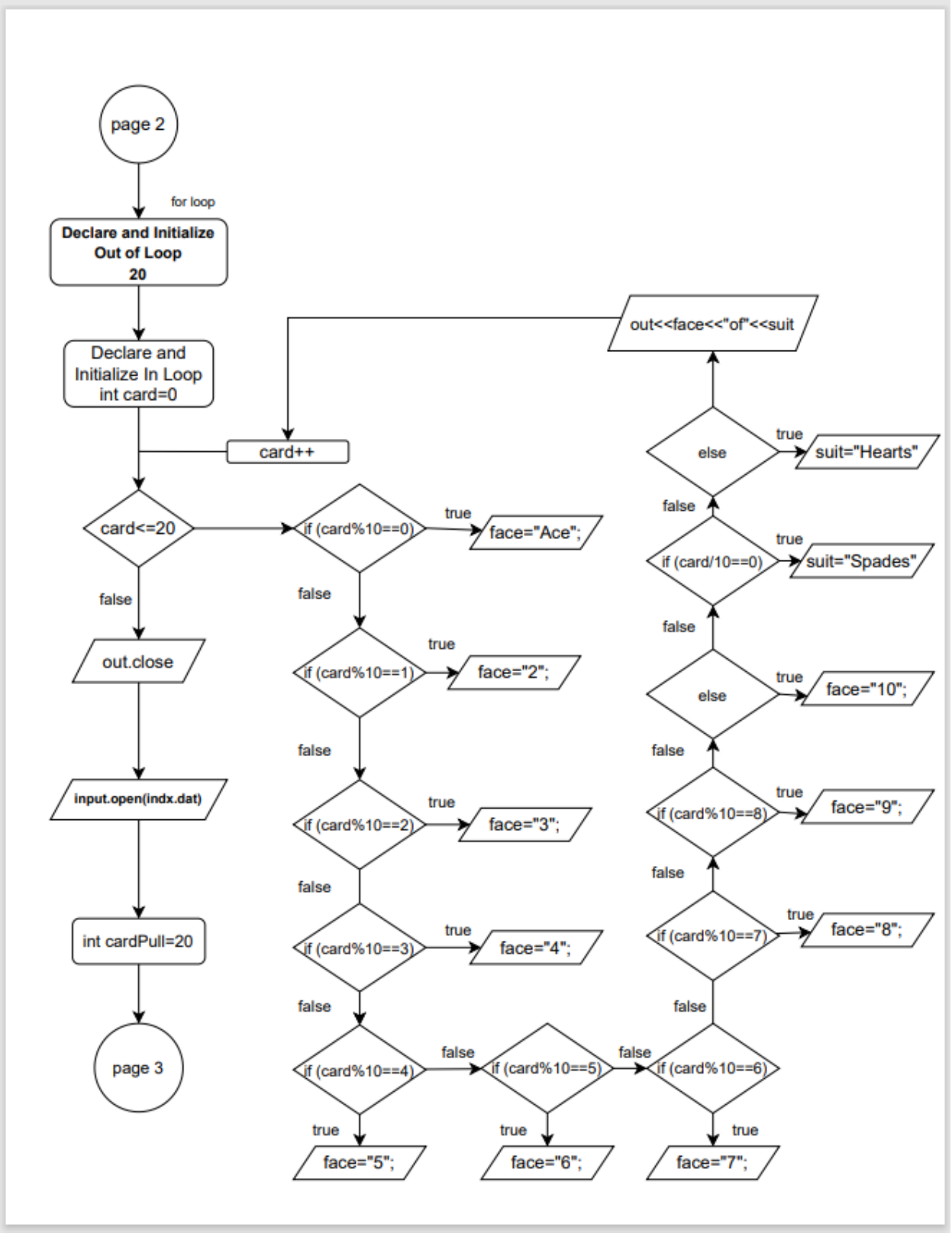
It was fun; if there were more time as a normal semester session, this project would have felt less constructed, but I did as much as I could within the week.

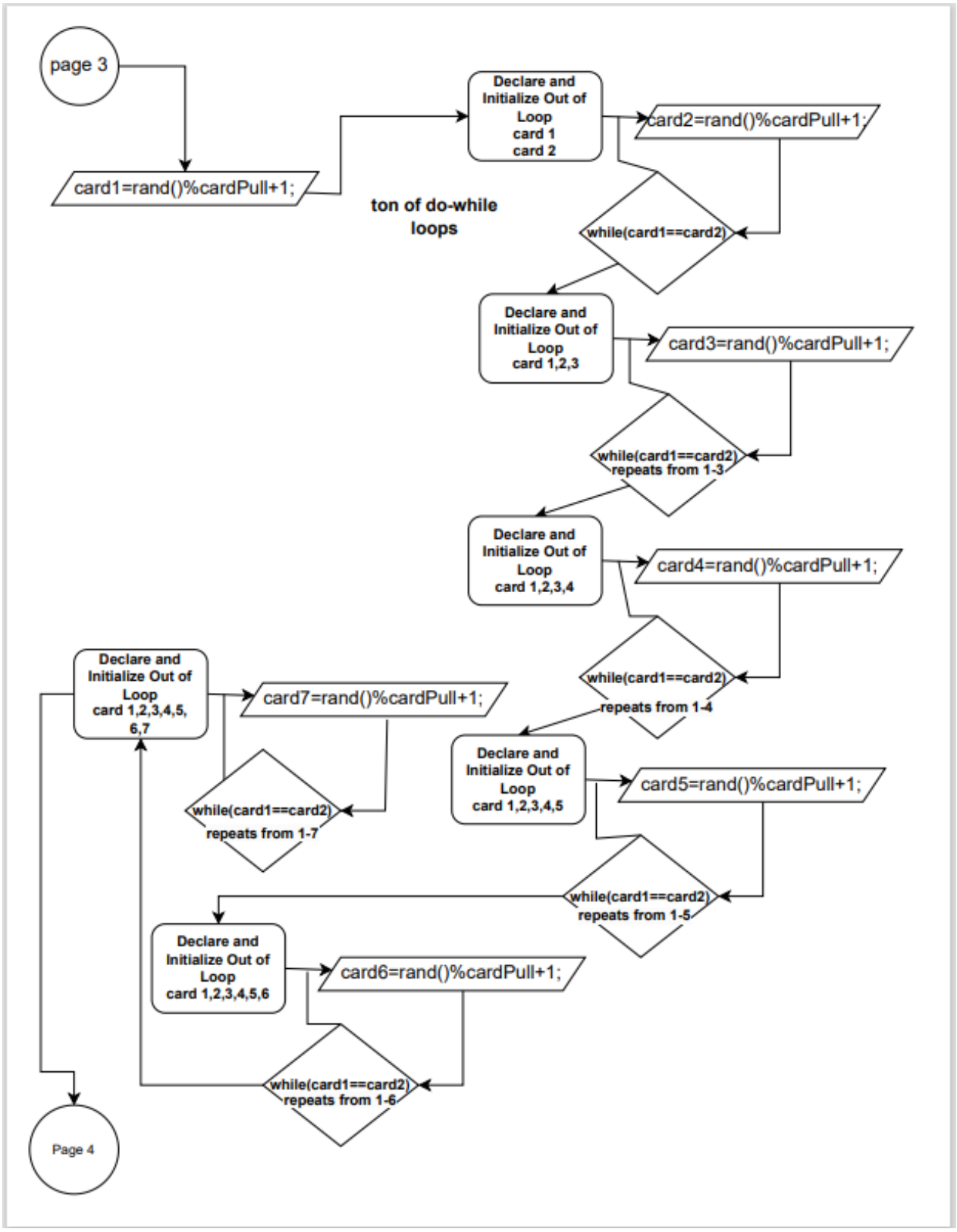
Description

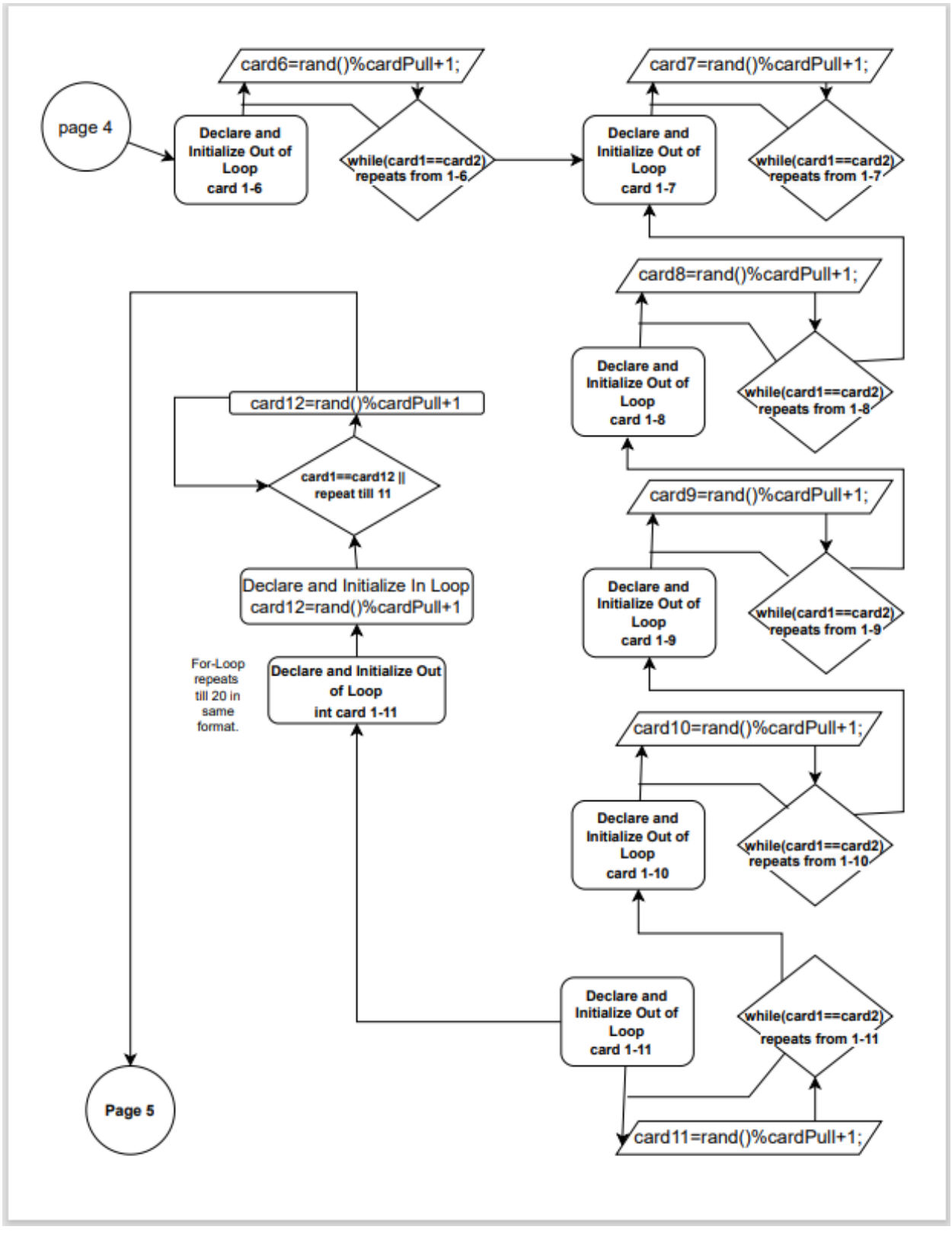
The main point is to match each letter to its' pair, in the least amount of moves possible; despite it not being fully a game of Concentration with the visuals, I wanted to implement the concept as best as I could.

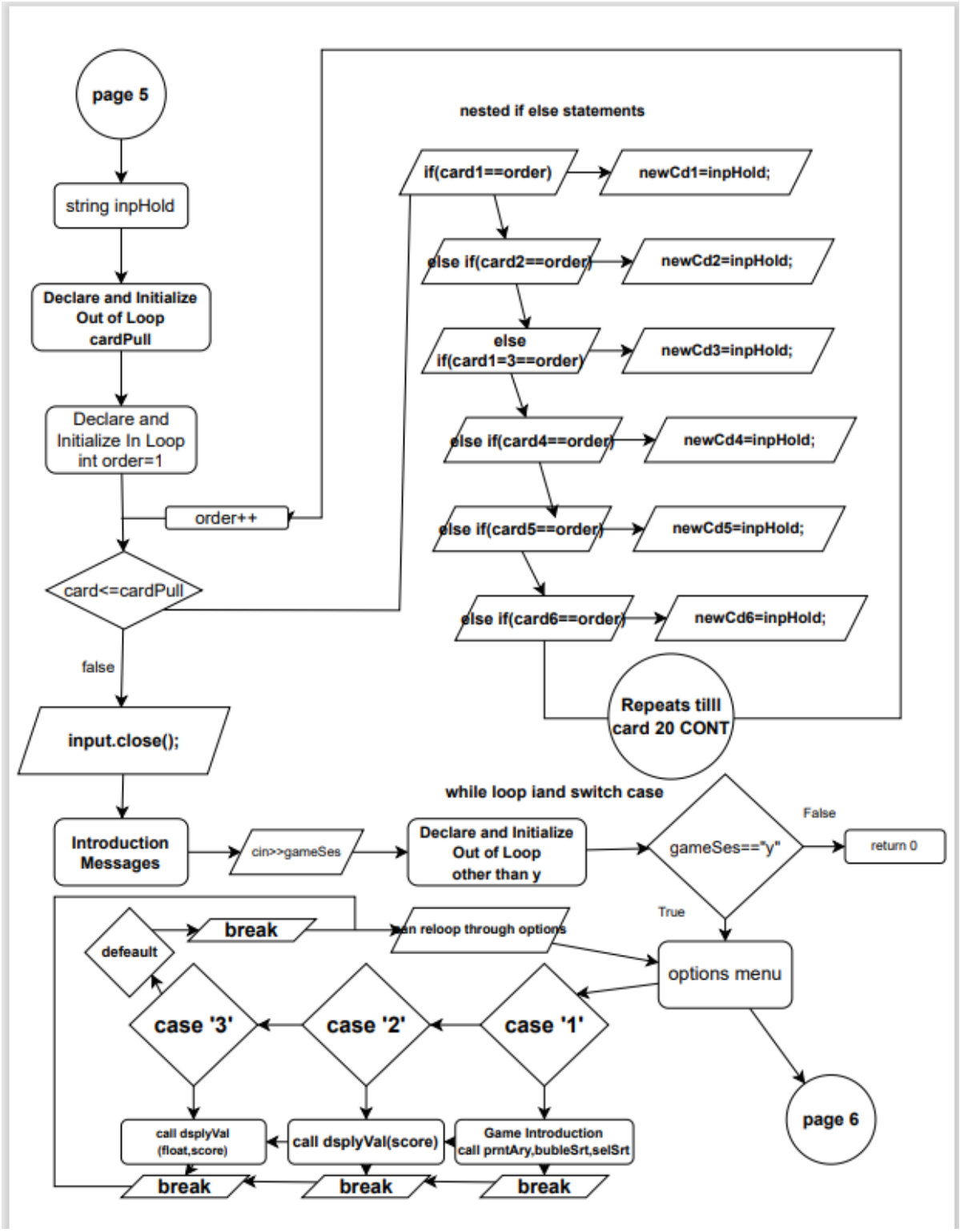
Flowchart (PAGES 1-11)

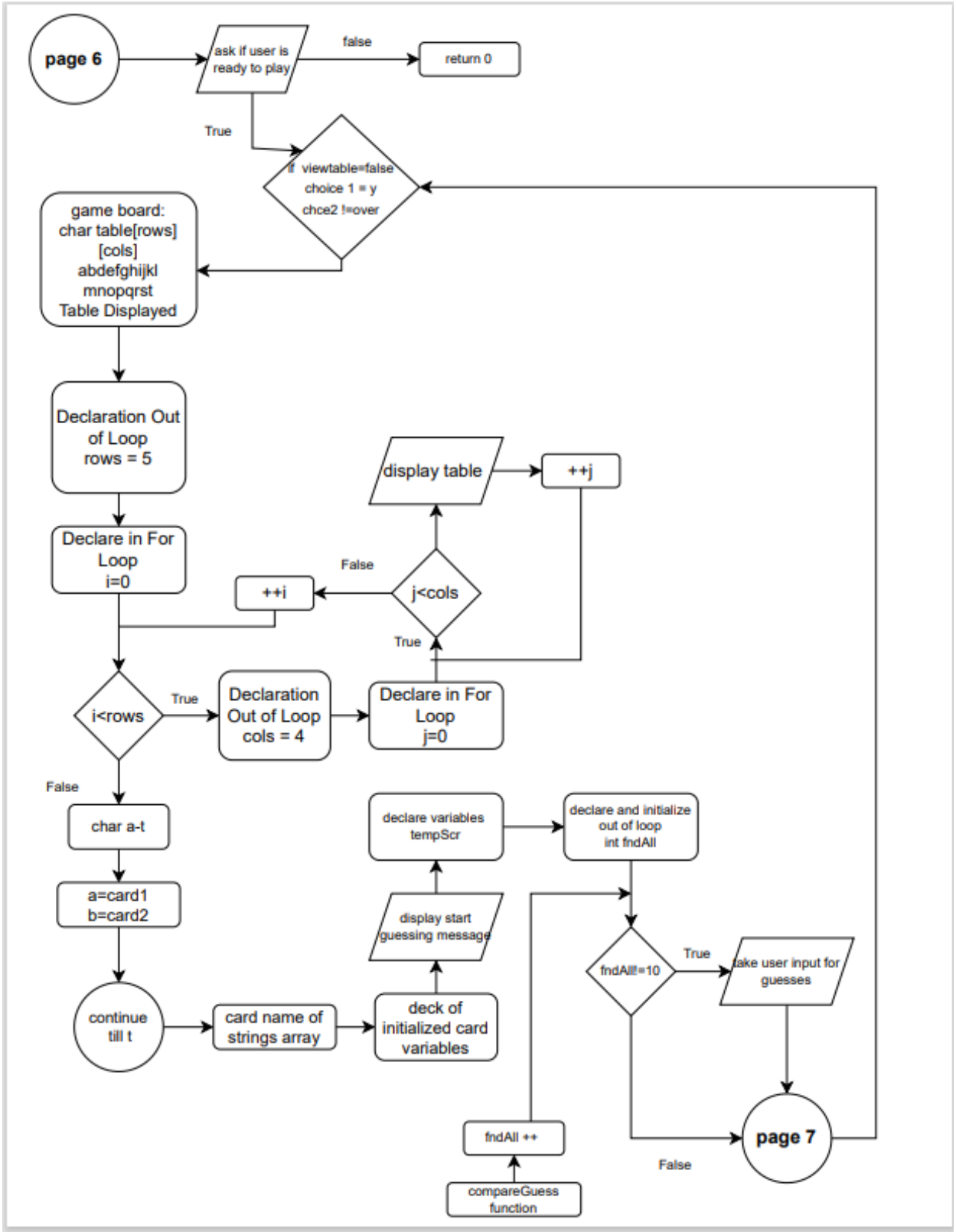


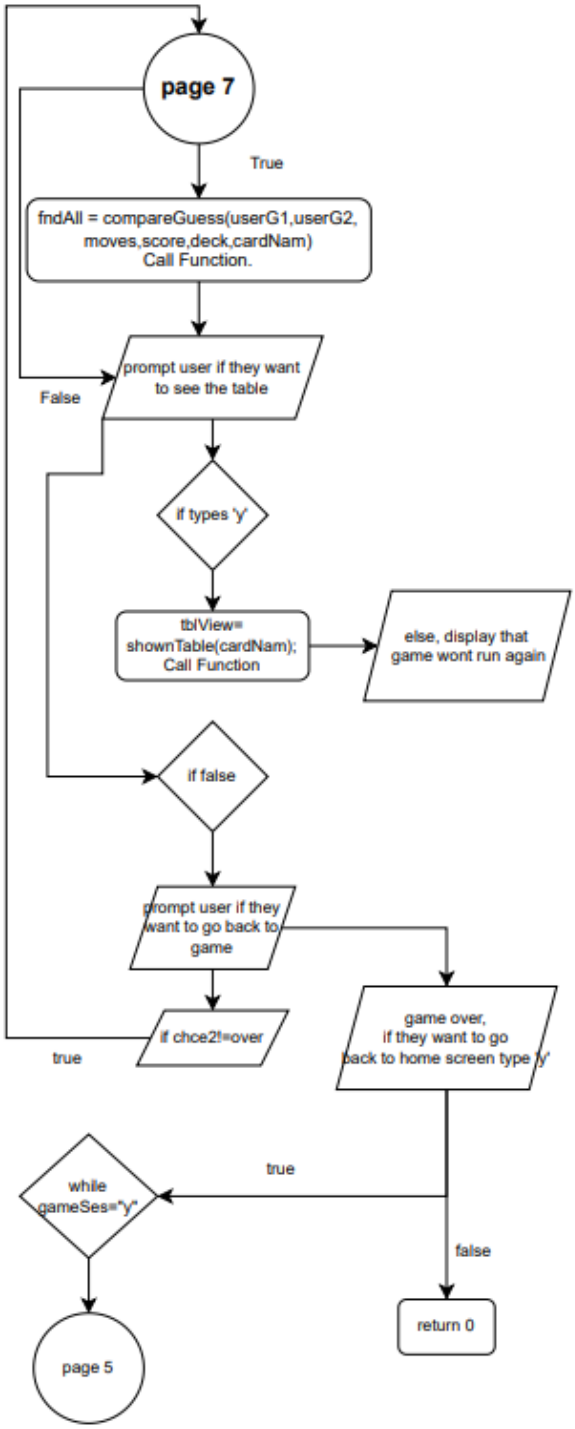

page 2
for loop
Declare and Initialize Out of Loop 20
Declare and Initialize In Loop int card=0
card++
card<=20
false
out.close
input.open(indx.dat)
int cardPull=20
page 3
if (card%10==0)
true
face="Ace";
false
if (card%10==1)
true
face="2";
false
if (card%10==2)
true
face="3";
false
if (card%10==3)
true
face="4";
false
if (card%10==4)
true
face="5";
false
if (card%10==5)
true
face="6";
false
if (card%10==6)
true
face="7";
false
if (card%10==7)
true
face="8";
false
if (card%10==8)
true
face="9";
false
else
true
face="10";
false
if (card/10==0)
true
suit="Spades"
false
else
true
suit="Hearts"
out<<face<<"of"<<suit

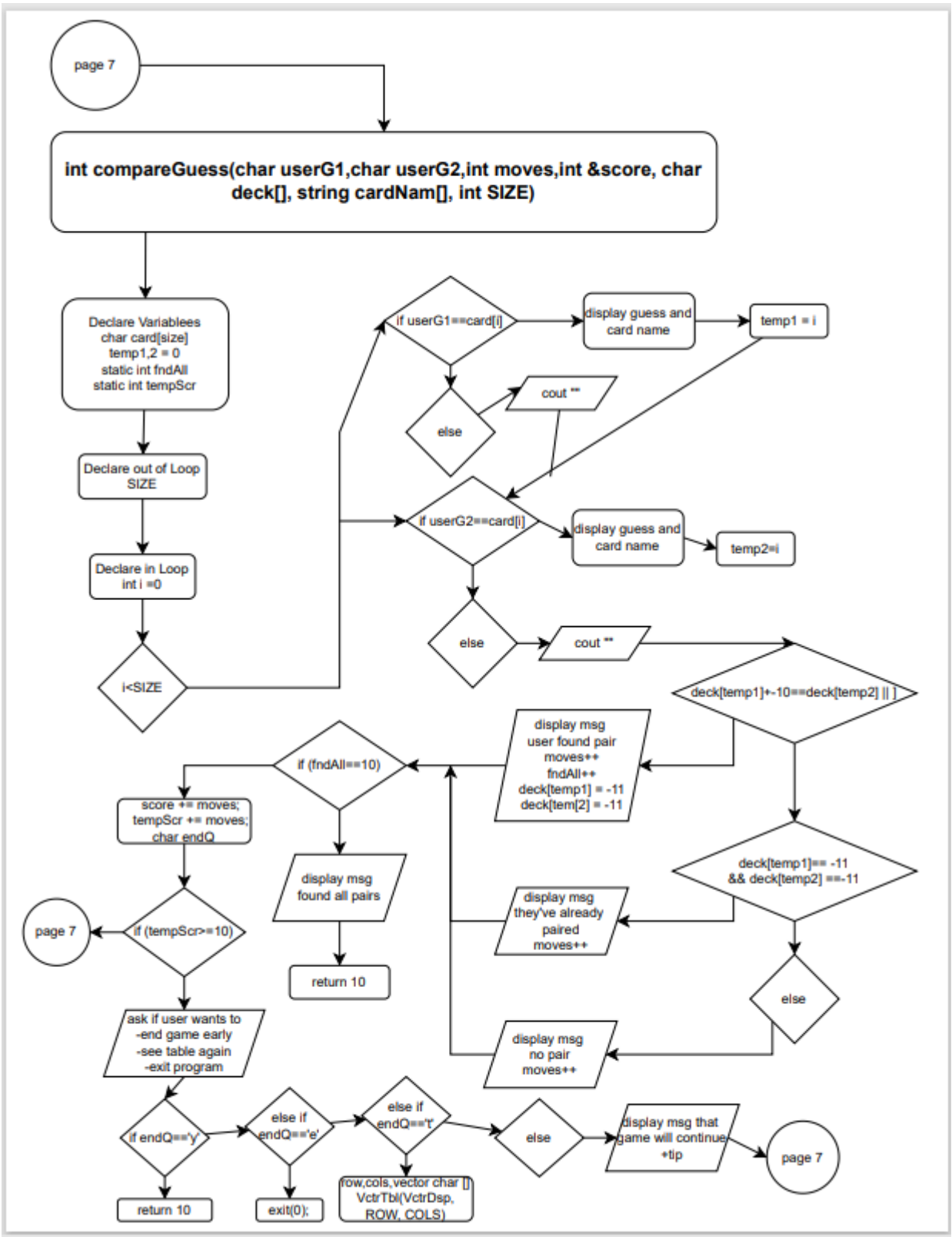

page 3
card1=rand()%cardPull+1;
Declare and Initialize Out of Loop card 1 card 2
card2=rand()%cardPull+1;
ton of do-while loops
while(card1==card2)
Declare and Initialize Out of Loop card 1,2,3
card3=rand()%cardPull+1;
while(card1==card2) repeats from 1-3
Declare and Initialize Out of Loop card 1,2,3,4
card4=rand()%cardPull+1;
while(card1==card2) repeats from 1-4
Declare and Initialize Out of Loop card 1,2,3,4,5,6,7
card7=rand()%cardPull+1;
while(card1==card2) repeats from 1-7
Declare and Initialize Out of Loop card 1,2,3,4,5
card5=rand()%cardPull+1;
while(card1==card2) repeats from 1-5
Declare and Initialize Out of Loop card 1,2,3,4,5,6
card6=rand()%cardPull+1;
while(card1==card2) repeats from 1-6
Page 4

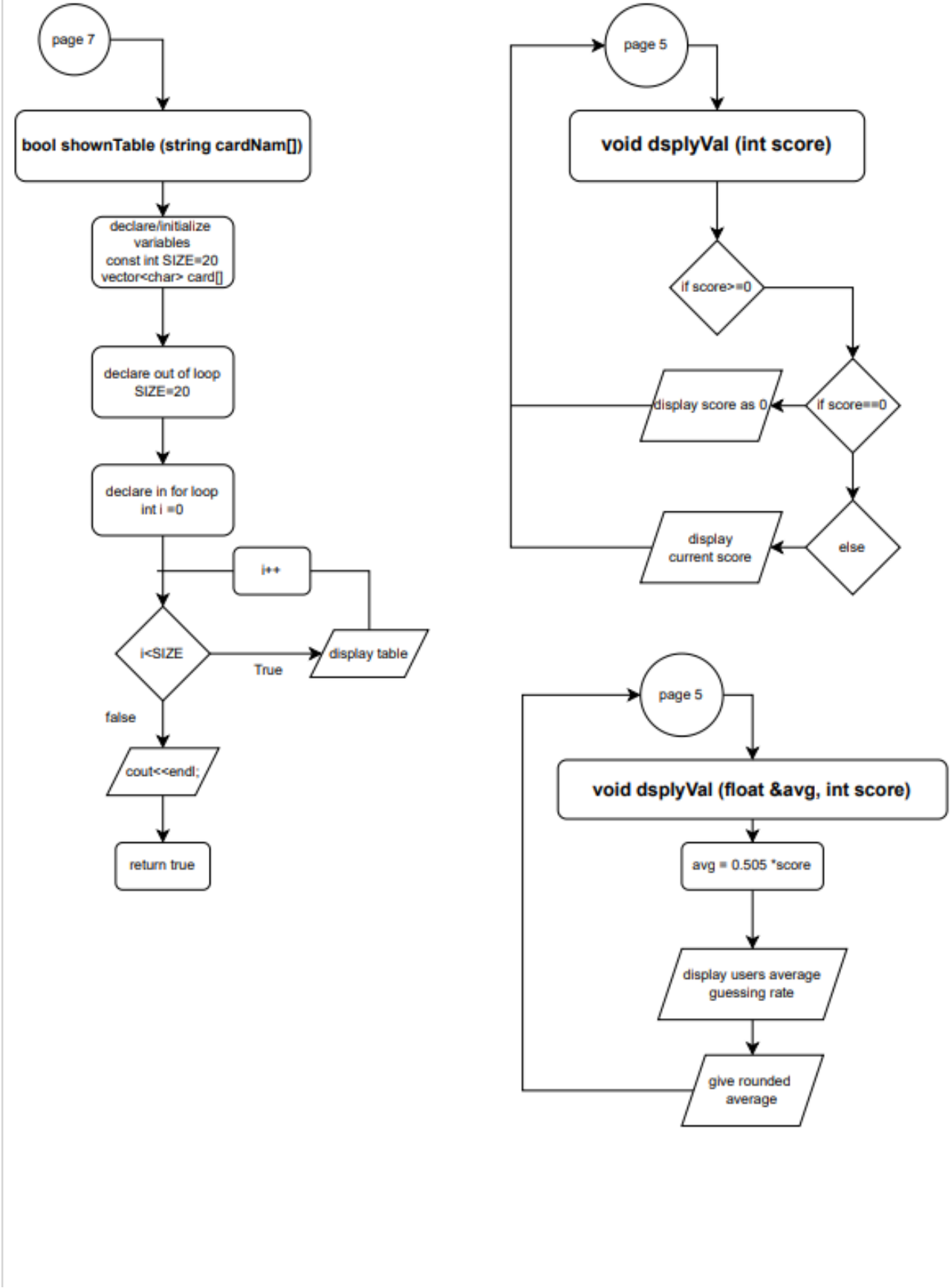

page 4
Declare and Initialize Out of Loop card 1-6
card6=rand()%cardPull+1;
while(card1==card2) repeats from 1-6
Declare and Initialize Out of Loop card 1-7
card7=rand()%cardPull+1;
while(card1==card2) repeats from 1-7
Declare and Initialize Out of Loop card 1-8
card8=rand()%cardPull+1;
while(card1==card2) repeats from 1-8
Declare and Initialize Out of Loop card 1-9
card9=rand()%cardPull+1;
while(card1==card2) repeats from 1-9
Declare and Initialize Out of Loop card 1-10
card10=rand()%cardPull+1;
while(card1==card2) repeats from 1-10
Declare and Initialize Out of Loop card 1-11
card11=rand()%cardPull+1;
while(card1==card2) repeats from 1-11
Declare and Initialize Out of Loop int card 1-11
For-Loop repeats till 20 in same format.
Declare and Initialize In Loop card12=rand()%cardPull+1
card12=rand()%cardPull+1
card1==card12 || repeat till 11
Page 5

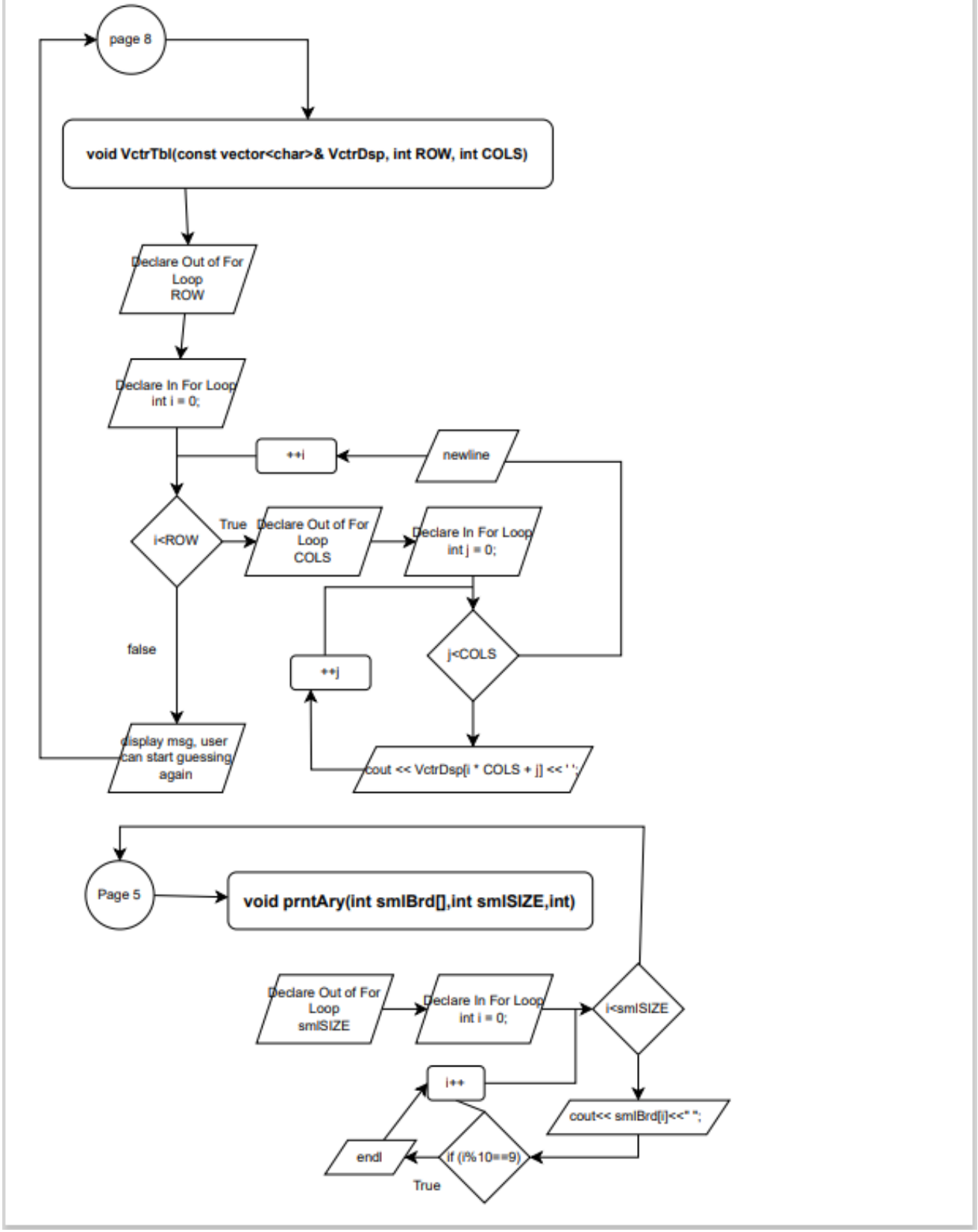

page 5
string inpHold
Declare and Initialize Out of Loop cardPull
Declare and Initialize In Loop int order=1
order++
card<=cardPull
false
input.close();
nested if else statements
if(card1==order)
newCd1=inpHold;
else if(card2==order)
newCd2=inpHold;
else if(card1=3==order)
newCd3=inpHold;
else if(card4==order)
newCd4=inpHold;
else if(card5==order)
newCd5=inpHold;
else if(card6==order)
newCd6=inpHold;
Repeats tilll card 20 CONT
while loop iand switch case
Introduction Messages
cin>>gameSes
Declare and Initialize Out of Loop other than y
gameSes=="y"
False
return 0
True
options menu
defeault
break
can reloop through options
case '3'
case '2'
case '1'
call dsplyVal (float,score)
call dsplyVal(score)
Game Introduction call prntAry,bubleSrt,selSrt
break
break
break
page 6

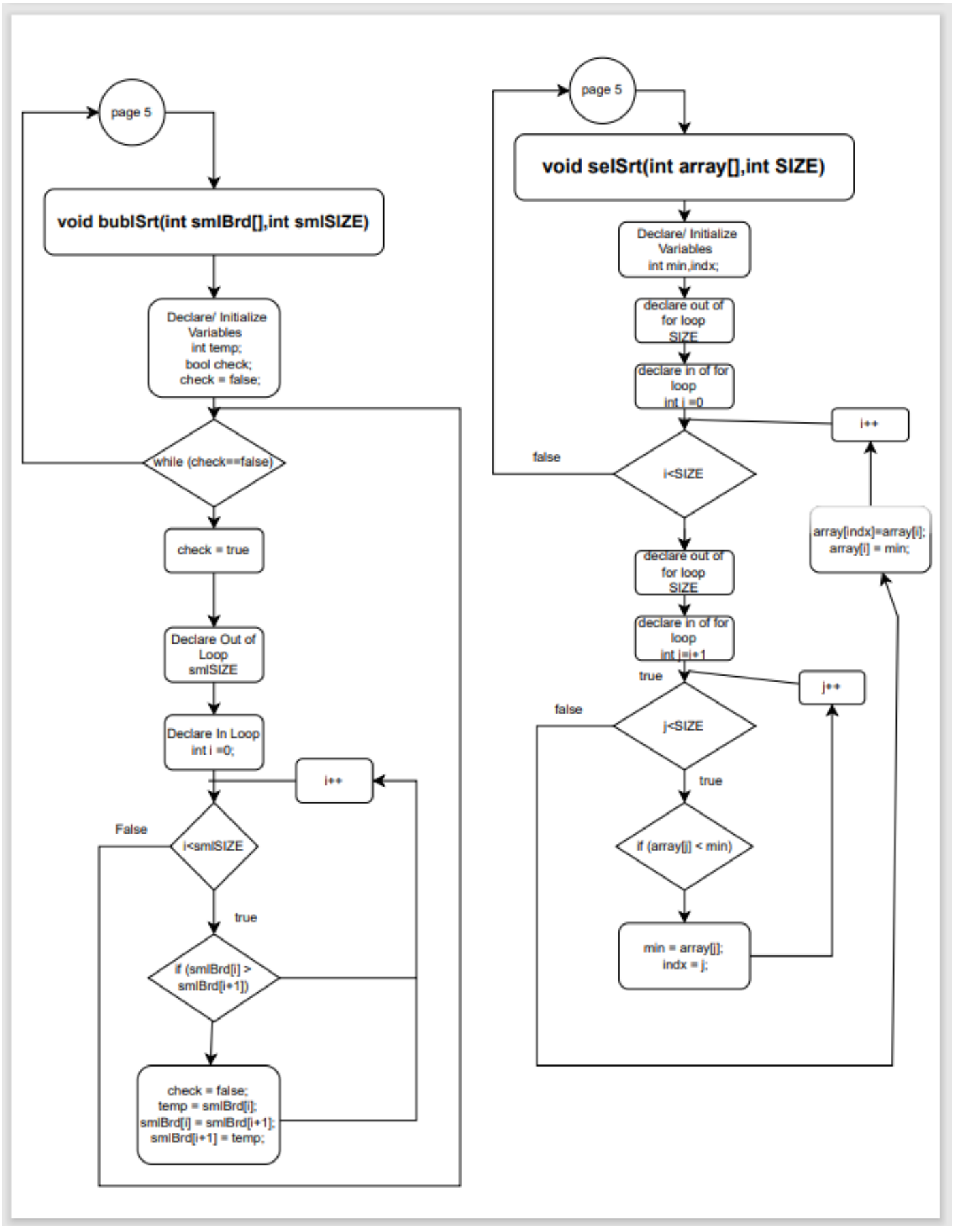

page 6
ask if user is ready to play
false
return 0
True
if viewtable=false
choice 1 = y
chce2 !=over
game board:
char table[rows]
[cols]
abdefghijkl
mnopqrst
Table Displayed
Declaration Out of Loop
rows = 5
Declare in For Loop
i=0
display table
++j
False
++i
j<cols
True
i<rows
True
Declaration Out of Loop
cols = 4
Declare in For Loop
j=0
False
char a-t
a=card1
b=card2
continue till t
card name of strings array
deck of initialized card variables
display start guessing message
declare variables
tempScr
declare and initialize out of loop
int fndAll
fndAll!=10
True
take user input for guesses
page 7
False
fndAll ++
compareGuess function


page 7
True
fndAll = compareGuess(userG1,userG2,
moves,score,deck,cardNam)
Call Function.
prompt user if they want
to see the table
False
if types 'y'
tblView=
shownTable(cardNam);
Call Function
else, display that
game wont run again
if false
prompt user if they
want to go back to
game
if chce2!=over
true
game over,
if they want to go
back to home screen type 'y'
while
gameSes="y"
true
false
return 0
page 5


page 7
int compareGuess(char userG1,char userG2,int moves,int &score, char deck[], string cardNam[], int SIZE)
Declare Variablees
char card[size]
temp1,2 = 0
static int fndAll
static int tempScr
Declare out of Loop
SIZE
Declare in Loop
int i =0
i<SIZE
if userG1==card[i]
display guess and card name
temp1 = i
else
cout ""
if userG2==card[i]
display guess and card name
temp2=i
else
cout ""
deck[temp1]+-10==deck[temp2] ||]
display msg
user found pair
moves++
fndAll++
deck[temp1] = -11
deck[tem[2] = -11
if (fndAll==10)
score += moves;
tempScr += moves;
char endQ
display msg
found all pairs
return 10
deck[temp1]== -11
&& deck[temp2] ==-11
display msg
they've already
paired
moves++
else
display msg
no pair
moves++
page 7
if (tempScr>=10)
ask if user wants to
-end game early
-see table again
-exit program
if endQ=='y'
return 10
else if
endQ=='e'
exit(0);
else if
endQ=='t'
row,cols,vector char []
VctrTbl(VctrDsp,
ROW, COLS)
else
display msg that
game will continue
+tip
page 7


page 7
bool shownTable (string cardNam[])
declare/initialize variables
const int SIZE=20
vector<char> card[]
declare out of loop
SIZE=20
declare in for loop
int i =0
i++
i<SIZE
True
display table
false
cout<<endl;
return true
page 5
void dsplyVal (int score)
if score>=0
if score==0
display score as 0
else
display current score
page 5
void dsplyVal (float &avg, int score)
avg = 0.505 *score
display users average guessing rate
give rounded average


page 8
void VctrTbl(const vector<char>& VctrDsp, int ROW, int COLS)
Declare Out of For Loop ROW
Declare In For Loop int i = 0;
++i
newline
i<ROW
True
Declare Out of For Loop COLS
Declare In For Loop int j = 0;
false
++j
j<COLS
display msg, user can start guessing again
cout << VctrDsp[i * COLS + j] << ' ';
Page 5
void prntAry(int smlBrd[],int smlSIZE,int)
Declare Out of For Loop smlSIZE
Declare In For Loop int i = 0;
i<smlSIZE
i++
cout<< smlBrd[i]<<" ";
endl
if (i%10==9)
True


page 5
void bublSrt(int smlBrd[],int smlSIZE)
Declare/ Initialize Variables
int temp;
bool check;
check = false;
while (check==false)
check = true
Declare Out of Loop
smlSIZE
Declare In Loop
int i =0;
i++
False
i<smlSIZE
true
if (smlBrd[i] > smlBrd[i+1])
check = false;
temp = smlBrd[i];
smlBrd[i] = smlBrd[i+1];
smlBrd[i+1] = temp;
page 5
void selSrt(int array[],int SIZE)
Declare/ Initialize Variables
int min,indx;
declare out of for loop
SIZE
declare in of for loop
int i =0
i++
false
i<SIZE
array[indx]=array[i];
array[i] = min;
declare out of for loop
SIZE
declare in of for loop
int j=i+1
true
j++
false
j<SIZE
true
if (array[j] < min)
min = array[j];
indx = j;

Pseudo Code

// Pseudocode for Concentration Card Game

/Import necessary libraries

Declare global constants and variables

Function Prototypes

FUNCTION compareGuess(CHAR userGuess1, CHAR userGuess2, INT moves, REF INT score, ARRAY deck[], ARRAY cardNames[], INT size)

FUNCTION shownTable(ARRAY cardNames[])

FUNCTION displayValue(INT score)

FUNCTION displayValue(FLOAT average, INT score)

FUNCTION vectorTable(VECTOR cardDisplay, INT rows, INT columns)

FUNCTION printArray(ARRAY smallBoard, INT size, INT columns)

FUNCTION bubbleSort(ARRAY arrayToSort, INT size)

FUNCTION selectionSort(ARRAY arrayToSort, INT size)

// Program Execution Begins Here

FUNCTION main()

  // Set a random seed

  seedRandom()

  // Declare variables

  ARRAY deck[deckSize] // Array to store card values

  ARRAY cardNames[deckSize] // Array to store card names (e.g., "5 of Hearts")

  VECTOR cardDisplay[deckSize] // Vector to represent the state of the card board

  BOOLEAN tableShown = FALSE // Flag to check if the table is revealed

```
    // Initialize card deck and names
    initializeDeck(deck)
    initializeCardNames(cardNames)

    // Display game introduction and options
    PRINT "Welcome to the Concentration Card Game!"
    PRINT "Would you like to enter the loading screen? (y/n)"
    READ gameSession

    WHILE gameSession = "y"
        // Display options menu
        displayOptionsMenu(hasPlayed)

        // Get user selection
        READ selection

        IF hasPlayed AND selection <> '4'
            PRINT "You've already played the game!"
        END IF

        SWITCH selection
            CASE '1':
                // Display introduction
Call functions to show visual examples
                displayIntroduction()
            CASE '2':
                // Display scores
```

```
        Call Functions to display Scores
            displayValue(score)
        CASE '3':
            // Display average
Call functions to display Averages
            displayValue(average, score)
        CASE '4':
            // Exit the program
            PRINT "Exiting the program..."
            RETURN 0
        DEFAULT:
            PRINT "Invalid option. Please try again."
    END SWITCH

    // If the user wants to play the game
    PRINT "Would you like to play the game? (y/any other character)"
    READ choice1

    // Convert uppercase to lowercase
    choice1 += IF choice1 <= 97 THEN 32 ELSE
    choice2 = ""
    WHILE (choice1 = 'y' AND choice2 <> "over") AND (choice1 = 'y' AND NOT
tableShown)
        // Display the Concentration Card Board
        displayCardBoard(cardDisplay, cardsPerRow, cardsPerColumn)

        // Get user guesses
        userGuess1, userGuess2 = getUserGuess()
```

```
        // Compare user guesses
        compareResult = compareGuess(userGuess1, userGuess2, moves, score, deck, cardNames, deckSize)

        IF compareResult = 10
            // User found all pairs
            PRINT "Congratulations! You've found all ten pairs."
        End

        // Display score and average
IF decided to go back to homescreen
        displayValue(score)
        displayValue(average, scores)

        // Ask user if they want to see the table
        PRINT "Would you like to see the table placements fully revealed? (type 'y' to view/else otherwise)"
        READ showTable

        IF showTable = 'y'
            // Show the table
            tableShown = shownTable(cardNames)
        END IF

        IF NOT tableShown
            // Ask user if they want to end the game
```

```
            PRINT "If you'd like to end the game, type 'over'. If you want to keep trying, type anything else."
            READ choice2
        ELSE
            PRINT "Since you've viewed the table, the game will not rerun."
        END IF
    END WHILE

    PRINT "If you'd like to go back to the home screen, type 'y'. If not, type any other character."
    hasPlayed = TRUE
    READ gameSession
  END WHILE

  // Exit the program
  PRINT "Exiting the program..."
  RETURN 0
END FUNCTION
```

Function to Compare Guess

Using the first and second users guess, takes their moves and the reference of the score, deck array, card name array and SIZE

COMPARE Cards less than 20 times

If users guess equals the letter in the alphabet, print the guess and cardName

If their guess is correct,

Display message that it is a pair,

Add a move

Assign the matched pairs a –11

ELSE IF the deck equals –11

Inform the user they guessed apart of a pair

ELSE

Inform the user there is no pair with their two guesses

If they found all 10

    Give a congratulatory message

Every 10 guesses, give them a few options

To end the game early, to end the program, or see the table again

Create a display table with the answers

Create a loop that prints vector char, which shows card to the string name of the card

Return, which now user cannot replay the game

Create two functions for displaying score and average

IF the score equals 0

Tell the user they need to play for it to update

ELSE

Tell their current score

Second function for the Average

Do simple math for 1/20 guess times score

Display their average guessing

    And it rounded.

Create a Vector Table

Rows and Cols in loops should display the table

Display message they can start guessing

Create another function that displays an array unsorted for the introducton

prints simulation board

Create a function doing Bubble Sort

WHILE check still equals False

Create loop that compares a number, to the one adjacent to i, right one larger, swaps numbers

And make sure to swap.

When all sorted make sure to set check to True to END WHILE

Create a function doing Selection Sort

Create a min and index

LOOP through unsorted areas and reassigns min to the smallest num compared to the previous min - found

index should hold the place of where the smallest num was found

Use the index and min to redefine places of the array.

CHECKLIST FOR PROJECT 1 & 2

Cross Reference from Project 1

You are to fill-in with where located in code

Chapter	Section	Topic	Where Line #'s	Pts	Notes
2	2	cout	262		
	3	libraries Declared at 9-16	Specifically Used At 241, 513, 516, 44, 69,44	5	iostream, iomanip, cmath, cstdlib, fstream, string, ctime USED VECTORS TOO.
	4	variables/literals	47-65		No variables in global area, failed project!
	5	Identifiers	47-66		
	6	Integers	404	1	
	7	Characters	335	1	
	8	Strings	359	1	
	9	Floats No Doubles	60	1	Using doubles will fail the project, floats OK!
	10	Bools	312	1	
	11	Sizeof *****			
	12	Variables 7 characters or less	47-63		All variables <= 7 characters
	13	Scope ***** No Global Variables			
	14	Arithmetic operators	514		
	15	Comments 20%+	everywhere within program	2	Model as pseudo code
	16	Named Constants	47		All Local, only Conversions/Physics/Math in Global area
	17	Programming Style ***** Emulate			Emulate style in book/in class repositiory
3	1	cin	257		
	2	Math Expression	514		
	3	Mixing data types ****			
	4	Overflow/Underflow ****			
	5	Type Casting	514	1	
	6	Multiple assignment *****			
	7	Formatting output		1	
	8	Strings	61	1	
	9	Math Library	516	1	All libraries included have to be used
	10	Hand tracing ******			
4	1	Relational Operators	521		
	2	if	535	1	Independent if

3	1	cin	257		
	2	Math Expression	514		
	3	Mixing data types ****			
	4	Overflow/Underflow ****			
	5	Type Casting	514	1	
	6	Multiple assignment *****			
	7	Formatting output		1	
	8	Strings	61	1	
	9	Math Library	516	1	All libraries included have to be used
	10	Hand tracing ******			
4	1	Relational Operators	521		
	2	if	535	1	Independent if
	4	If-else	501,504	1	
	5	Nesting	460,463,466,473	1	
	6	If-else-if	71,74,77	1	
	7	Flags *****			
	8	Logical operators	431	1	
	11	Validating user input	460,463,466	1	
	13	Conditional Operator	310	1	
	14	Switch	260	1	
5	1	Increment/Decrement	521	1	
	2	While	370	1	
	5	Do-while	259	1	
	6	For loop	533	1	
	11	Files input/output both	69,119	2	
	12	No breaks in loops ******			Failed Project if included
****** Not required to show			Total	30	

Cross Reference for Project 2

You are to fill-in with where located in code

Chapter	Section	Topic	Where Line #"s	Pts	Notes
6		Functions	372		
	3	Function Prototypes	34,36,38	4	Always use prototypes
	5	Pass by Value	24	4	
	8	return	443	4	A value from a function
	9	returning boolean	495	4	
	10	Global Variables		XXX	Do not use global variables -100 pts
	11	static variables	405	4	
	12	defaulted arguments	24	4	
	13	pass by reference	24	4	
	14	overloading	28/30	5	
	15	exit() function	464	4	
7		Arrays	402		
	1 to 6	Single Dimensioned Arrays	362	3	
	7	Parallel Arrays	359 & 362	2	
	8	Single Dimensioned as Function Arguments	372	2	
	9	2 Dimensioned Arrays	318	2	Emulate style in book/in class repositiory
	12	STL Vectors	469	2	
		Passing Arrays to and from Functions	24	5	
		Passing Vectors to and from Functions	32	5	
8		Searching and Sorting Arrays			
	3	Bubble Sort	541	4	
	3	Selection Sort	558	4	
	1	Linear or Binary Search	407 Linear Search	4	
		*chose to do linear search			
****** Not required to show			Total	70	Other 30 points from Proj 1 first sheet tab

Reference

1. Textbook

2. Lehr's GitHub Repository

3. Concentration Game Introduction/How to Play

4. Project 1

Versions (Project Folder holds 7 Versions (5-11))

Version 5: Originally, I started with the contents of project 1 to have the basis for this project. I would have chosen a different game or something a bit more complex, but with the few days within the winter intersession, I had to make do. Granted, making a concentration game was not as difficult as I thought once we learned arrays and functions. Within the first version, I mainly worked on figuring out what functions I could implement within my code. I started with what the main component of the game would be about, the guessing portion. Within the first version of Project 2 (Version 5) I created the compare Guess function.

Version 7: Within this version, I was able to successfully create the comparison between the guesses. My main function was to have the users know whether they've guessed a pair yet or if their guesses weren't paired. After, I was able to implement the statement for if the user found all 10 pairs. There wasn't as much done within this version, as I was mainly focused on getting the main function finished. I sat most of the time figuring out what to implement instead of writing, however there are more fixes within areas. It was a lot of trial and error and deleting portions I thought were useless.

Version 9: Most of the work was tanked within these last few versions of 9-11. I also spent the longest time within these versions trying to nitpick at what could be implemented. Since I do not have game graphics knowledge, I wanted to make the text seem as game-like as possible with a home screen and a surplus of user decisions. In this version, the user is now able to end the game early if they choose to. I created a vector version table to display the answers of the cards, next to its letters. A ton of this portion was bulk focusing on utilizing each requirement off the checklist.

Version 11: Version 11 was my last and final version with the time I had left. The main components of the program were finished by now, but there were a few items off the checklist I needed to complete. It was the selection sort and bubble sort, which I implemented within the beginning of the switch case. (In the introduction area). I added a

bit more to the compare guess's function. Regarding the user getting a pop up every 10 guesses, they were given the opportunity to choose if they would prefer to continue, see the table, exit the whole program, or exit the game portion. I skipped each of the versions by a few to talk about the main differences made with each version. Compared to version 10, this holds most comments and refinements in comparison. All in all, I felt that there was more I could do, however, for my first somewhat finished project in C++, I am somewhat glad. Learning the language itself within a couple weeks was not easy, however the focus and attention on the main details within lectures aided the time constraints.

Tips/Advice for my Past Self,

1. Write out the main concepts and breakdown the needs for your game.
2. Work from portion to portion.
3. Look at the requirements needed then code.
4. Reference requirements often while coding.
5. Start working on tiny bits each day, even if you only come up with one idea.
6. Spend a day mapping out a game structure

Input and Output Testcases

Run 1:

```
The Concentration Card Board!

  _____________________
  | a   | b   | c   | d |
  | e   | f   | g   | h |
  | i   | j   | k   | l |
  | m   | n   | o   | p |
  | q   | r   | s   | t |
  _____________________
This is your board, try to find the matching pair to a!
Type two letters to see if you found a match! (exa. a g)
ab
a was 9ofSpades.
b was 7ofSpades.
No pair..

tr
r was 3ofHearts.
t was 10ofHearts.
No pair..

vg
g was 5ofHearts.
No pair..

lk
k was 4ofHearts.
l was 10ofSpades.
No pair..

as
a was 9ofSpades.
s was 4ofSpades.
No pair..

ar
a was 9ofSpades.
r was 3ofHearts.
No pair..

ab
a was 9ofSpades.
b was 7ofSpades.
No pair..

ac
a was 9ofSpades.
c was 9ofHearts.
They are a pair!

ad
a was 9ofSpades.
d was AceofSpades.
No pair..
```

```
Would you like to end the game early? (type 'y' to end)
If you'd like to end the program itself (type 'e' to terminate)
If you'd like to see the table again (type 't' to view table)
Type else if otherwise.

y

Would you like to see the table placements fully revealed? (type 'y' to view/else otherwise)
Disclaimer : Viewing the table means you aren't allowed to replay the game pass this point!
y
        a - 9ofSpades          b - 7ofSpades          c - 9ofHearts          d - AceofSpades          e - 6ofSpades
        f - 3ofSpades          g - 5ofHearts          h - 2ofSpades          i - 7ofHearts          j - 8ofHearts
        k - 4ofHearts          l - 10ofSpades          m - 6ofHearts          n - 2ofHearts          o - 5ofSpades
        p - 8ofSpades          q - AceofHearts          r - 3ofHearts          s - 4ofSpades          t - 10ofHearts

Since, you've viewed the table the game will not rerun!

If you'd like to go back to the home screen type y, if not, any other character will do.
```

Run 2:

```
 _____________________
| a | b | c | d |
| e | f | g | h |
| i | j | k | l |
| m | n | o | p |
| q | r | s | t |
 _____________________
This is your board, try to find the matching pair to a!
Type two letters to see if you found a match! (exa. a g)
ac
a was 9ofSpades.
c was 9ofHearts.
They are a pair!

fr
f was 3ofSpades.
r was 3ofHearts.
They are a pair!

ks
k was 4ofHearts.
s was 4ofSpades.
They are a pair!

p
j
j was 8ofHearts.
p was 8ofSpades.
They are a pair!

bigotlqdachnfrdqbihnknempjoglt
b was 7ofSpades.
i was 7ofHearts.
They are a pair!

g was 5ofHearts.
o was 5ofSpades.
They are a pair!

l was 10ofSpades.
t was 10ofHearts.
They are a pair!

d was AceofSpades.
q was AceofHearts.
They are a pair!

a was 9ofSpades.
c was 9ofHearts.
These have already been matched to a pair..!

h was 2ofSpades.
n was 2ofHearts.
They are a pair!

You've found all ten pairs! Congrats.
```

Run 3:

```
If you'd like to go back to the home screen type y, if not, any other character will do. y
_____________________________
 Welcome to the Options Menu!
Option One (type 1) : Introduction.
Option Two (type 2) : Scores.
Option Three (type 3) : Average.
_____________________________

Enter a Character to Skip Options Screen.
You've played the game already!
2


Your score is currently 19.

Would you like to select another option? (type 1,2,3/ if not any type any other character.)
3

Your average guessing rate is currently 0.95.
Rounded, it is 1.00!


Would you like to select another option? (type 1,2,3/ if not any type any other character.)
1

A half a stack of cards, only the suit of Hearts and Spades will be shuffled (excluding Joker and Royals).
20 Cards (Ace-9) will be laid out individually and the goal is to match two cards in the least amount of turns..
A matching pair is defined as two cards with the same rank. (two Aces, two fives, etc.)
If the two cards make a pair, you take them and count that as a point. This is a memory game!
Project 2's UPDATE, means you must guess all 10 pairs! ( Previously, project 1 had you try to find the matching card for letter a, instead of all 20.)
Since there isn't a visual stack of cards, the user must type a placement (exa. a b, f a), after choosing, the cards will be revealed.

The board will be technically set up this way,
20 11 3 13 5 19 17 8 16 14
2 12 4 10 15 9 1 18 6 7


and present itself sorted this way,
1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20

Another visual example with a different shuffle could be,
12 10 20 1 5 6 13 7 2 9
11 4 8 14 19 16 17 18 15 3


However, sorted with pairs, it would imitate this,
1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20


Each pair is scrambled, the way 1/10 are and 2/11 are.

Would you like to select another option? (type 1,2,3/ if not any type any other character.)
```

Program

```
/*
* File:   main.cpp
* Author: Emma Wuysang
* Created on February 10, 2024, 2:23 PM
* Purpose: Concentration Card Game V.11 (Final Version)
*/

//System Libraries
#include <iostream>  //I/O Library
#include <cstdlib>   //Random Function Library
#include <ctime>     //Time Library
#include <iomanip>   //Formatting Library
#include <fstream>   //File Library
#include <string>    //String Objects
#include <cmath>     //Math Library
#include <vector>   // Vector Library
using namespace std; //Library Name-space

//User Libraries

//Global Constants - Math,Physics,Chemistry,Conversions

//Function Prototypes
int compareGuess(char,char,int,int &, char[], string[], int SIZE=20); //compares the two guesses, main portion of the game
```

```
bool shownTable (string[]); //display table after leaving guessing portion - shows answers

void dsplyVal(int score); // displays score, made for function overloading

void dsplyVal(float &avg, int score); // displays average, made for function overloading

void VctrTbl(const vector<char>& VctrTbl, int ROW, int COLS); // vector table version, redisplayed by user in guessing

void prntAry(int [],int,int); // prints within switch case, shows an example

void bublSrt(int [],int); // prints within switch case, shows an example, sorts by bubble sort

void selSrt(int [],int); // prints within switch case, shows an example, sorts by selection sort

//Program Execution Begins Here
int main(int argc, char** argv) {

  //Set a random seed
  srand(static_cast<unsigned int>(time(0))); // seeds w/ current time.

  //Declare all variables
  const int cardAmt=20; // Amount of Cards
  char chce1; // choice 1
  string chce2; // choice 2
  fstream out; // output file
  string fileName; // opens fstream
```

```
   string face, suit; // Hearts and Spades + Ace-10
   unsigned short card1, card2, card3, card4, card5, card6, card7, card8, card9, card10,
                  card11,card12,card13,card14,card15,card16,card17,card18,card19,card20; // group of
cards
   fstream input; // input file
   string
newCd1,newCd2,newCd3,newCd4,newCd5,newCd6,newCd7,newCd8,newCd9,newCd10,

newCd11,newCd12,newCd13,newCd14,newCd15,newCd16,newCd17,newCd18,newCd19,newC
d20; // takes the file, sets random sections to string.
   char slctn; // selection
   int score = 0.00; // general score
   float avg = 0.00; // guessing rate
   string gameSes; // go to loading screen/continue game
   bool ran = false; // initializes
   int smlSIZE=20; // used for the board displays
   int smlBrd[]= {12,10,20,1,5,6,13,7,2,9,11,4,8,14,19,16,17,18,15,3}; // one unsorted array for
example 2
   int smlBrd1[]= {20,11,3,13,5,19,17,8,16,14,2,12,4,10,15,9,1,18,6,7}; // one unsorted array for
example 1

   //Process or Map solutions
   fileName="Indx.dat"; // card Index
   out.open(fileName,ios::out);
   for (int card=0;card<=20;card++){
      if (card%10==0){ // Ace
         face = "Ace";
      }
      else if (card%10==1){ // Two
```

```
   face = "2";
}
else if (card%10==2){ // Three
   face = "3";
}
else if (card%10==3){ // Four
   face = "4";
}
else if (card%10==4){ // Five
   face = "5";
}
else if (card%10==5){ // Six
   face = "6";
}
else if (card%10==6){ // Seven
   face = "7";
}
else if (card%10==7){ // Eight
   face = "8";
}
else if (card%10==8){ // Nine
   face = "9";
}
else{ // Ten
  face = "10";
}
```

```
   // Spades Appending Suit
   if (card/10==0){
      suit="Spades";
   }
   // Hearts Appending Suit
   else{
      suit="Hearts";
   }

   out<<face<<"of"<<suit<<endl;
}

// close the file
out.close();

// file name
fileName="Indx.dat";
input.open(fileName.c_str(),ios::in);

//Initialize Variables
int cardPull=20;

//Unique Value for card 1.
card1=rand()%cardPull+1;
do{
   card2=rand()%cardPull+1;
```

```
  //Unique Value for card 1,2.
  }while(card1==card2);

  //Unique Value for card 1,2,3.
  do{
    card3=rand()%cardPull+1;
  }while(card1==card3 || card2==card3);

  //Unique Value for card 1,2,3,4.
  do{
    card4=rand()%cardPull+1;
  }while(card1==card4 || card2==card4 || card3==card4);

  //Unique Value for card 1,2,3,4,5.
  do{
    card5=rand()%cardPull+1;
  }while(card1==card5 || card2==card5 || card3==card5 || card4==card5);

  //Unique Value for card 1,2,3,4,5,6.
  do{
    card6=rand()%cardPull+1;
  }while(card1==card6 || card2==card6 || card3==card6 || card4==card6 || card5==card6);

  //Unique Value for card 1,2,3,4,5,6,7.
  do{
    card7=rand()%cardPull+1;
  }while(card1==card7 || card2==card7 || card3==card7 || card4==card7 || card5==card7 ||
card6==card7);
```

```
  //Unique Value for card 1,2,3,4,5,6,7,8.
  do{
    card8=rand()%cardPull+1;
  }while(card1==card8 || card2==card8 || card3==card8 || card4==card8 || card5==card8 ||
card6==card8 || card7==card8);

  //Unique Value for card 1,2,3,4,5,6,7,8,9.
  do{
    card9=rand()%cardPull+1;
  }while(card1==card9 || card2==card9 || card3==card9 || card4==card9 || card5==card9 ||
card6==card9 || card7==card9 || card8==card9);

  //Unique Value for card 1,2,3,4,5,6,7,8,9,10.
  do{
    card10=rand()%cardPull+1;
  }while(card1==card10 || card2==card10 || card3==card10 || card4==card10 || card5==card10 ||
card6==card10 || card7==card10 || card8==card10 || card9==card10);

  //Unique Value for card 1,2,3,4,5,6,7,8,9,10,11.
  do{
    card11=rand()%cardPull+1;
  }while(card1==card11 || card2==card11 || card3==card11 || card4==card11 || card5==card11 ||
card6==card11 || card7==card11 || card8==card11 || card9==card11 || card10==card11);

  //Unique Value for card 1,2,3,4,5,6,7,8,9,10,11,12 using for loops.
  for(card12=rand()%cardPull+1;card1==card12 || card2==card12 || card3==card12 ||
card4==card12 || card5==card12 || card6==card12 || card7==card12 || card8==card12 ||
card9==card12 || card10==card12 || card11==card12;card12=rand()%cardPull+1);
```

```
   //Unique Value for card 1,2,3,4,5,6,7,8,9,10,11,12,13 using for loops.

   for(card13=rand()%cardPull+1;card1==card13 || card2==card13 || card3==card13 ||
card4==card13 || card5==card13 || card6==card13 || card7==card13 || card8==card13 ||
card9==card13 || card10==card13 || card11==card13 ||
card12==card13;card13=rand()%cardPull+1);


   //Unique Value for card 1,2,3,4,5,6,7,8,9,10,11,12,13,14 using for loops.

   for(card14=rand()%cardPull+1;card1==card14 || card2==card14 || card3==card14 ||
card4==card14 || card5==card14 || card6==card14 || card7==card14 || card8==card14 ||
card9==card14 || card10==card14 || card11==card14 || card12==card14 ||
card13==card14;card14=rand()%cardPull+1);


   //Unique Value for card 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 using for loops.

   for(card15=rand()%cardPull+1;card1==card15 || card2==card15 || card3==card15 ||
card4==card15 || card5==card15 || card6==card15 || card7==card15 || card8==card15 ||
card9==card15 || card10==card15 || card11==card15 || card12==card15 || card13==card15 ||
card14==card15 ;card15=rand()%cardPull+1);


   //Unique Value for card 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16 using for loops.

   for(card16=rand()%cardPull+1;card1==card16 || card2==card16 || card3==card16 ||
card4==card16 || card5==card16 || card6==card16 || card7==card16 || card8==card16 ||
card9==card16 || card10==card16 || card11==card16 || card12==card16 || card13==card16 ||
card14==card16 || card15==card16;card16=rand()%cardPull+1);


   //Unique Value for card 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17 using for loops.

   for(card17=rand()%cardPull+1;card1==card17 || card2==card17 || card3==card17 ||
card4==card17 || card5==card17 || card6==card17 || card7==card17 || card8==card17 ||
card9==card17 || card10==card17 || card11==card17 || card12==card17 || card13==card17 ||
card14==card17 || card15==card17 || card16==card17;card17=rand()%cardPull+1);


   //Unique Value for card 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18 using for loops.
```

```
   for(card18=rand()%cardPull+1;card1==card18 || card2==card18 || card3==card18 ||
card4==card18 || card5==card18 || card6==card18 || card7==card18 || card8==card18 ||
card9==card18 || card10==card18 || card11==card18 || card12==card18 || card13==card18||
card14==card18 || card15==card18 || card16==card18 ||
card17==card18;card18=rand()%cardPull+1);


   //Unique Value for card 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,19 using for loops.

   for(card19=rand()%cardPull+1;card1==card19 || card2==card19 || card3==card19 ||
card4==card19 || card5==card19 || card6==card19 || card7==card19 || card8==card19 ||
card9==card19 || card10==card19 || card11==card19 || card12==card19 || card13==card19 ||
card14==card19 || card15==card19 || card16==card19 || card17==card19||
card18==card19;card19=rand()%cardPull+1);


   //Unique Value for card 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,19,20 using for loops.

   for(card20=rand()%cardPull+1;card1==card20 || card2==card20 || card3==card20 ||
card4==card20 || card5==card20 || card6==card20 || card7==card20 || card8==card20 ||
card9==card20 || card10==card20 || card11==card20 || card12==card20 || card13==card20 ||
card14==card20 || card15==card20 || card16==card20 || card17==card20|| card18==card20 ||
card19==card20 ;card20=rand()%cardPull+1);



   //Pulls cards from the file

   string inpHold; // input holder


   /*loops from 1-20, taking the random num and finding

    *  the position of the string, finally,

    * setting into a variable.

    */


   for(int order=1;order<=cardPull;order++){ // like stated above, this assigns the cards

      input>>inpHold;
```

```
    if(card1==order)newCd1=inpHold;
    else if(card2==order)newCd2=inpHold;
    else if(card3==order)newCd3=inpHold;
    else if(card4==order)newCd4=inpHold;
    else if(card5==order)newCd5=inpHold;
    else if(card6==order)newCd6=inpHold;
    else if(card7==order)newCd7=inpHold;
    else if(card8==order)newCd8=inpHold;
    else if(card9==order)newCd9=inpHold;
    else if(card10==order)newCd10=inpHold;
    else if(card11==order)newCd11=inpHold;
    else if(card12==order)newCd12=inpHold;
    else if(card13==order)newCd13=inpHold;
    else if(card14==order)newCd14=inpHold;
    else if(card15==order)newCd15=inpHold;
    else if(card16==order)newCd16=inpHold;
    else if(card17==order)newCd17=inpHold;
    else if(card18==order)newCd18=inpHold;
    else if(card19==order)newCd19=inpHold;
    else if(card20==order)newCd20=inpHold;
}
input.close(); // closes the file after taking in the variables



//Display the output
cout<<"This the Concentration Card Game!\n";
```

```
  cout<<"Would you like to enter to the loading screen?(y/n) \n";
  cin>>gameSes;

  while(gameSes=="y"){ // leads to loading screen
    cout<<"_____\n";
    cout<<setw(30)<<"Welcome to the Options Menu!\n"
      <<setw(31)<<"Option One (type 1) : Introduction. \n"
      <<setw(30)<<"Option Two (type 2) : Scores.\n"
      <<setw(30)<<"Option Three (type 3) : Average.\n"
      <<"_____\n"
      <<endl<<setw(30)<<"Enter a Character to Skip Options Screen.\n";

    if (ran){
      cout<<"You've played the game already!\n";
    }
    cin>>slctn; // selection
    cout<<endl;
    do{ // loops the switch case for user to go through as many times.
      switch(slctn){
        case '1': // basic introduction
          cout<<endl<<"A half a stack of cards, only the suit of Hearts and Spades will be shuffled (excluding Joker and Royals). \n"
            <<"20 Cards (Ace-9) will be laid out individually and the goal is to match two cards in the least amount of turns.. \n"
            <<"A matching pair is defined as two cards with the same rank. (two Aces, two fives, etc.)  \n"
            <<"If the two cards make a pair, you take them and count that as a point. This is a memory game!\n"
```

```
        <<"Project 2's UPDATE, means you must guess all 10 pairs! ( Previously, project 1 had you try to find the matching card for letter a, instead of all 20.) \n"
        <<"Since there isn't a visual stack of cards, the user must type a placement (exa. a b, f a), after choosing, the cards will be revealed.\n"<<endl;



        cout<<"The board will be technically set up this way, \n";
        prntAry(smlBrd1,smlSIZE,10); // prints first unsorted array
        cout<<endl;
        selSrt(smlBrd1,smlSIZE); // selection sorts array
        cout<<endl;
        cout<<"and present itself sorted this way,\n";
        prntAry(smlBrd1,smlSIZE,10); // prints first array sorted
        cout<<endl;
        cout<<"Another visual example with a different shuffle could be, \n";



        prntAry(smlBrd,smlSIZE,10); // prints second array unsorted
        cout<<endl;
        cout<<endl<<"However, sorted with pairs, it would imitate this, \n";
        bublSrt(smlBrd,smlSIZE); // bubble sorts array
        prntAry(smlBrd,smlSIZE,10); // prints array sorted
        cout<<endl;
        cout<<endl<<"Each pair is scrambled, the way 1/10 are and 2/11 are. \n";
        cout<<endl<<"Would you like to select another option? (type 1,2,3/ if not any type any other character.)\n";
        cin>>slctn;
        break;
```

```
            case '2': // scores
               dsplyVal(score);
               cout<<endl<<"Would you like to select another option? (type 1,2,3/ if not any type any other character.)\n";
               cin>>slctn; break;

            case '3':// average
               dsplyVal(avg,score);
               cout<<endl<<"Would you like to select another option? (type 1,2,3/ if not any type any other character.)\n";
               cin>>slctn;
               break;
            default:break;   //ends case '3'
         }
      }while(slctn=='1' || slctn=='2' || slctn=='3'); // selection if equals 1,2,or 3.
      // If the user wants to play the game
      cout<<"Want To Play?(y/any other character) \n"<<endl; // user input to begin game
      cin>>chce1;
      cout<<endl<<endl<<endl;

      chce1+=(chce1<=97)?32:0; // uppercase to lowercase
      chce2="";
      bool tblView = false;
      while((chce1 == 'y' && chce2 != "over") && (chce1 == 'y' && tblView == false)){ // if the user hasn't declared over, or hasn't viewed the answer table
         cout<<setw(28)<<"The Concentration Card Board!\n"; // initial game display
         const int rows = 5; // initialized for two dimensional array
```

```
const int cols = 4;
cout<<setw(26)<<"_____\n";
char table[rows][cols] = { // two dimensional array table display
  {'a', 'b', 'c', 'd'},
  {'e', 'f', 'g', 'h'},
  {'i', 'j', 'k', 'l'},
  {'m', 'n', 'o', 'p'},
  {'q', 'r', 's', 't'} };

// displays the table
for (int i=0; i < rows; ++i) {
  for (int j=0; j < cols; ++j) {
    cout<<setw(4)<< "| " <<table[i][j]<<" ";
  }
  cout<<"|\n";
}

cout<<setw(26)<<"_____\n";

char a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t, userG1, userG2; // for comparison from string letter to char
int SIZE = 20;
a=card1; //takes value from initial card1 (etc)
b=card2;
c=card3;
d=card4;
e=card5;
f=card6;
```

```
        g=card7;
        h=card8;
        i=card9;
        j=card10;//takes value from initial card10 (etc)
        k=card11;
        l=card12;
        m=card13;
        n=card14;
        o=card15;
        p=card16;
        q=card17;
        r=card18;
        s=card19;
        t=card20;//takes value from initial card20 (etc)

        // parallel arrays for card letter to string name
        string cardNam[20] =
{newCd1,newCd2,newCd3,newCd4,newCd5,newCd6,newCd7,newCd8,newCd9,newCd10,

newCd11,newCd12,newCd13,newCd14,newCd15,newCd16,newCd17,newCd18,newCd19,newC
d20}; //holds all the names of the cards in string

        char deck[SIZE] = {a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t}; // holds the initialized deck

        cout<<"This is your board, try to find the matching pair to a! \n"
            "Type two letters to see if you found a match! (exa. a g)\n";
        unsigned int moves = 0; // counter for amount of moves made
```

```
        int fndAll = 0;
        int tempScr = 0;
        while(fndAll != 10){ //stops when pair is found
          cin>>userG1>>userG2; // users guess 1 and 2
          fndAll = compareGuess(userG1,userG2,moves,score,deck,cardNam); // fndAll determines if they've found all 10 pairs.
        }

        char shwTbl;
        cout<<endl<<"Would you like to see the table placements fully revealed? (type 'y' to view/else otherwise) \n" // shows table answers, won't allow you to play the game anymore
            <<"Disclaimer : Viewing the table means you aren't allowed to replay the game pass this point! \n";
        cin>>shwTbl;
        if (shwTbl=='y'){ // shows table if wanted
          tblView=shownTable(cardNam);
        }
        if (tblView == false){ // if table is chosen not to be viewed
          cout<<endl<<"If you'd like to end the game, type over, if you want to keep trying type anything else. \n";
          cout<<"Reminder, to get a fresh shuffled set, please re-run the program! \n";
          cin>>chce2;    // ends loop of game, can get better score if tried again.
        }
        else{
          cout<<endl<<"Since, you've viewed the table the game will not rerun! \n"<<endl;
        }
      }
      cout<<"If you'd like to go back to the home screen type y, if not, any other character will do. ";
      ran=true; // if user has run the game
```

```
    cin>>gameSes; // ends program if any other character then y typed.
  }
  //Exit the program
  return 0;
}



int compareGuess(char userG1,char userG2,int moves,int &score, char deck[], string cardNam[], int SIZE){ // compares the guesses


  char card[SIZE] = {'a','b','c','d','e','f','g','h','i', // made to loop through and check if guess is equal
    'j','k','l','m','n','o','p','q','r','s','t'};
  int temp1,temp2 = 0; // holds  indices to access deck
  static int fndAll = 0;
  static int tempScr = 0;
  for (int i=0;i<SIZE;i++){ // iterates through the card size, does linear search through array for specific card
    if (userG1==card[i]){ // if their guess 1 is inside the ASCII array of a-t
      cout<<userG1<<" was "<<cardNam[i]<<". "<<endl;
      temp1 = i;
    }
    else{
      cout<<"";
    }
    if (userG2==card[i]){ // if their guess 2 is inside the ASCII array of a-t
      cout<<userG2<<" was "<<cardNam[i]<<". "<<endl;
      temp2 = i;
    }
```

```
    else{
      cout<<"";
    }
  }

  if (deck[temp1]+10==deck[temp2]|| deck[temp1]-10==deck[temp2] ){ // check if the two guesses are a pair
    cout<<"They are a pair! \n"<<endl;
    moves += 1;
    fndAll+=1;
    deck[temp1] = -11; // accounts for if you've guessed something or not, cannot re-guess the same pair
    deck[temp2] = -11;
  }
  else if (deck[temp1]==-11 && deck[temp2]==-11){ // if user guesses the same pair or one from two pairs
    cout<<"These have already been matched to a pair..!\n"<<endl;
    moves += 1;
  }
  else{
    cout<<"No pair.."<<endl; // message if the two guesses weren't a pair
    cout<<endl;
    moves += 1;
  }

  if (fndAll==10){ // when user has found all ten pairs
    cout<<"You've found all ten pairs! Congrats.\n"<<endl;
    return 10;
```

```
  }

  score += moves;
  tempScr += moves;
  char endQ; // decision to end game early

  if (tempScr>10){ // this is so the temporary score resets each time
    tempScr = 0;
  }

  if (tempScr>=10){
    cout<<"Would you like to end the game early? (type 'y' to end) \n"
        "If you'd like to end the program itself (type 'e' to terminate) \n"
        "If you'd like to see the table again (type 't' to view table) \n"
        "Type else if otherwise. \n"<<endl;
    cin>>endQ;
    if (endQ=='y'){ // if user decides on to end game early
      return 10;
    }
    else if (endQ=='e'){ // if user decides to exit the entire program
      exit(0);
    }
    else if (endQ=='t'){ // if user decides to see the table again
      const int ROW = 5;
      const int COLS = 4;
      vector<char> VctrDsp = {'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t'};  // the vector
display table
```

```
        VctrTbl(VctrDsp, ROW, COLS);
      }
      else{
        cout<<"The game will continue!\n"<<endl;
        cout<<"Tip : Writing down your guesses could come in handy! \n"<<endl; // gives tip to user after 10 guesses, and if they decide to continue playing
      }
    }
}

bool shownTable (string cardNam[]){ // option to display table after leaving guessing portion update
    const int SIZE = 20;
    vector<char> card = {'a','b','c','d','e','f','g','h','i',
                         'j','k','l','m','n','o','p','q','r','s','t'};

    for (int i=0;i<SIZE;i++){ // prints vector char, which shows card to the string name of the card
      cout<<setw(10);
      cout << card[i] << " - " << cardNam[i] << "  ";
      if (i==4 || i==9 || i ==14){ // newline
        cout<<endl;
      }
    }

    cout<<endl;

    return true; // you've seen the table! cannot keep playing.
}
```

```
void dsplyVal (int score){ // displays the scores

   if (score>=0){
      if (score==0){ // starting score
         cout<<endl<<"Your score is currently 0.00 . This will update once you play. \n";
      }
      else{ // after run through score
         cout<<endl<<"Your score is currently "<<score<<".\n";
      }
   }

}

void dsplyVal(float &avg, int score){ // displays averages

   cout<<fixed<<setprecision(2)<<showpoint; // iomanip library use
   avg = 0.05 *static_cast<float>(score); // 0.05 from 1/20
   cout<<endl<<"Your average guessing rate is currently "<<avg<<"."<<endl
      <<"Rounded, it is "<<round(avg)<<"!\n"<<endl; // utilizing round from cmath library
}

void VctrTbl(const vector<char>& VctrDsp, int ROW, int COLS) {
   // prints vector table
   for (int i = 0; i < ROW; ++i) { //iterates through rows
      for (int j = 0; j < COLS; ++j) { // iterates through columns
         cout << VctrDsp[i * COLS + j] << ' '; // displays vector table
      }
```

```
      cout << '\n';
   }
   cout<<endl;
   cout<<"You can start guessing again!\n";
}



void prntAry(int smlBrd[],int smlSIZE,int){ // prints the array
   for (int i=0;i<smlSIZE;i++){
      cout<< smlBrd[i]<<" "; // prints simulation board
      if (i%10==9){
         cout<<endl;
      }
   }
}

void bublSrt(int smlBrd[],int smlSIZE){ // does bubble sort
   int temp; // holds value
   bool check;
   check = false;
   while (check==false){
      check = true; // means that nothing swapped, ending while loop
      for (int i=0;i<smlSIZE-1;i++){
         if (smlBrd[i] > smlBrd[i+1]){ // compares a number, to the one adjacent to i, right one larger,
swaps numbers
            check = false; // continues the while loop
            temp = smlBrd[i];
            smlBrd[i] = smlBrd[i+1];
```

```
            smlBrd[i+1] = temp;
          }
        }
      }
    }

void selSrt(int array[],int SIZE){ // performs the selection sort
    int min,indx;
    for (int i=0;i<SIZE-1;i++){
        min = array[i];
        indx = i;
        for (int j=i+1;j<SIZE;j++){ // makes sure to compare unsorted areas only
            if (array[j] < min){
                min = array[j]; // reassigns min to the smallest num compared to the previous min -  found
                indx = j; // this holds the place of where the smallest num was found

            }
        }
        array[indx]=array[i]; // puts first element in smaller element
        array[i] = min; // puts smaller element in front
    }
}
```