Project 2 Chopsticks

Bryanna Phan
Dr. Lehr
CSC-5 46091
07/29/15

Table of Contents

Introduction:

Welcome to the game of Chopsticks. This is a popular children's game often played during recess. Other popular names for this game are Sword, Sticks, and Magic Fingers.

Directions to Play:

To play, we first start off with both partners holding out one finger on each hand. You often pick someone to go first. In my program, the user will go first. Next, you tap one of your opponent's fingers. Your opponent must then hold out one additional finger on the hand you tapped (for a total of two) because the hand you used to tap them with had one finger held out. Let your opponent tap your hand. If they tap you with the hand that has one finger held out, you must hold out one additional finger on your hand that they tapped (totaling two). If they tap you with their hand that has two fingers held out, then you must add two fingers to your hand that they tapped (totaling three). Keep taking turns tapping hands and adding fingers, but when a hand has five fingers held out, that is called a "dead hand". Put dead hands behind the player's back. The person who reaches two dead hands first loses.

Project Summary:

- **Lines of Code:** ~488 lines
- **Number of Variables:**
    - char - 5
    - short - 4
    - vector - 1
    - int - 5
    - float - 1
    - string - 3
    - ofstream - 1
    - fstream - 1
- **Number of Functions: 5 + main**
- **Number of System Libraries: 5**

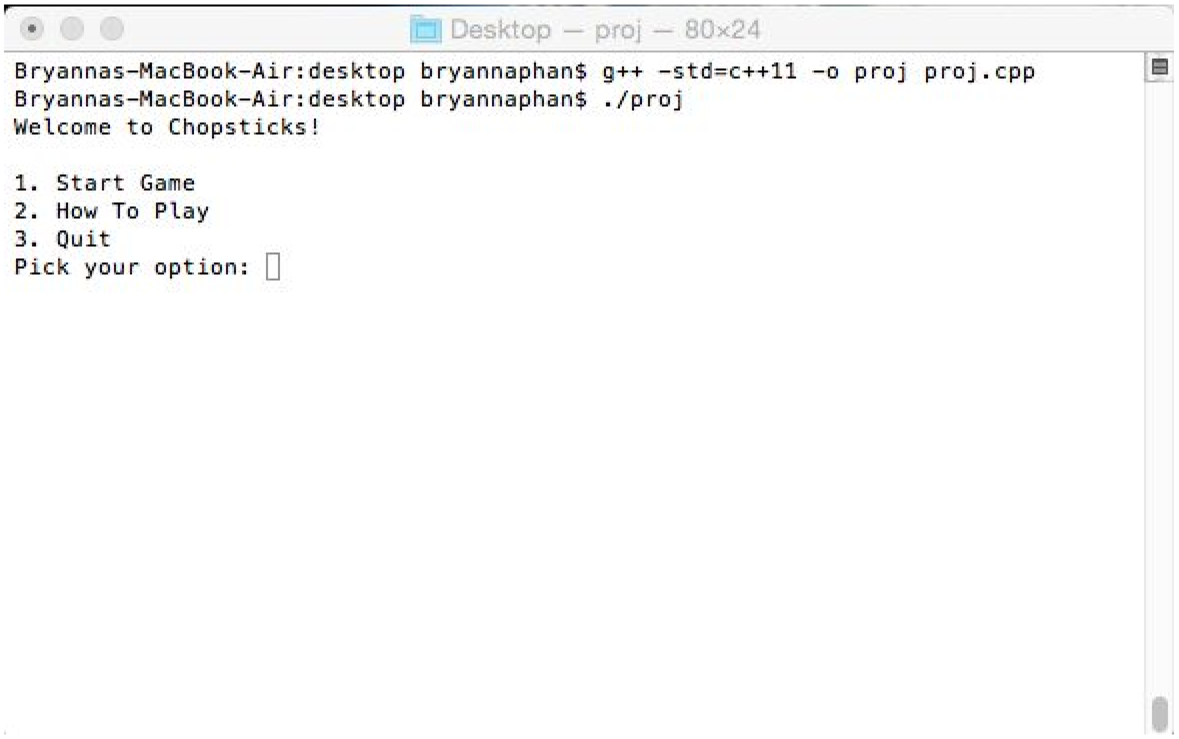
Thoughts:

I believe that I chose a very good game for our level of C++. However, there are still many limitations with this program. I was able to fix some of these limitations by utilizing arrays to represent all four of the hands.

Concepts I've implemented into my program are a switch statement, i/o files, void functions, for-loops, while loops, if-else statements, nested loops, cin/cout statements, different variables, "tolower" function, and different orders of operations using the (+=) shortcuts we learned, bool functions, bool statements, arrays, structures, and break statements.

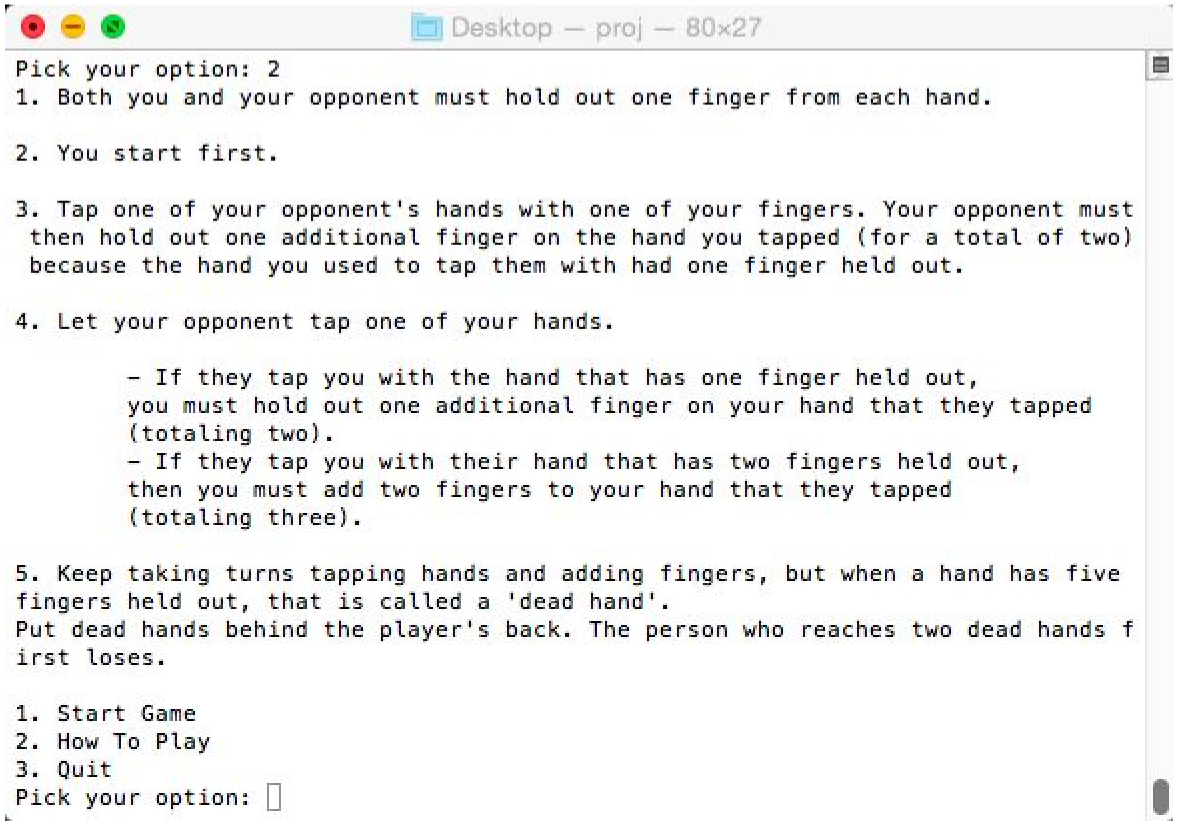
Running the Program

1. **Output Menu**
   - User has a choice to pick whether to start game, read instructions or quit



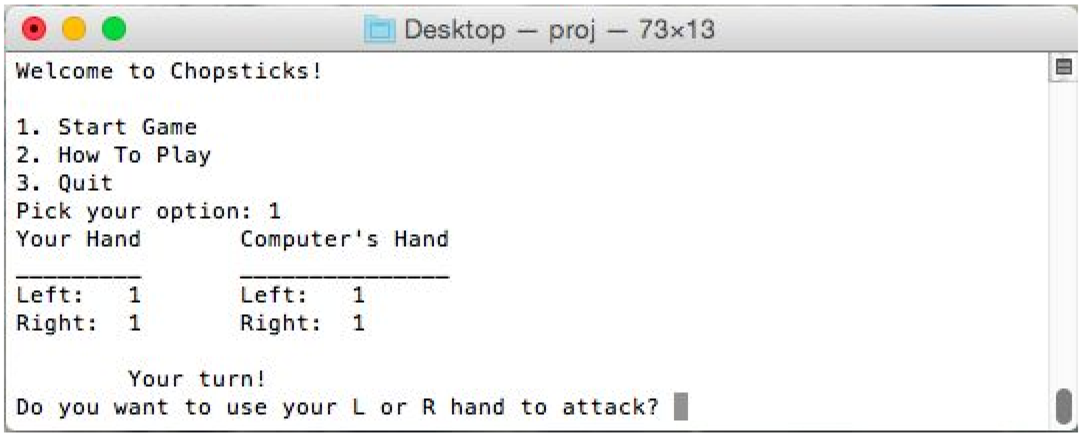
```
Bryannas-MacBook-Air:desktop bryannaphan$ g++ -std=c++11 -o proj proj.cpp
Bryannas-MacBook-Air:desktop bryannaphan$ ./proj
Welcome to Chopsticks!

1. Start Game
2. How To Play
3. Quit
Pick your option: 
```

2. **To view instructions**
   - The function prints out the instructions and then reprints the menu



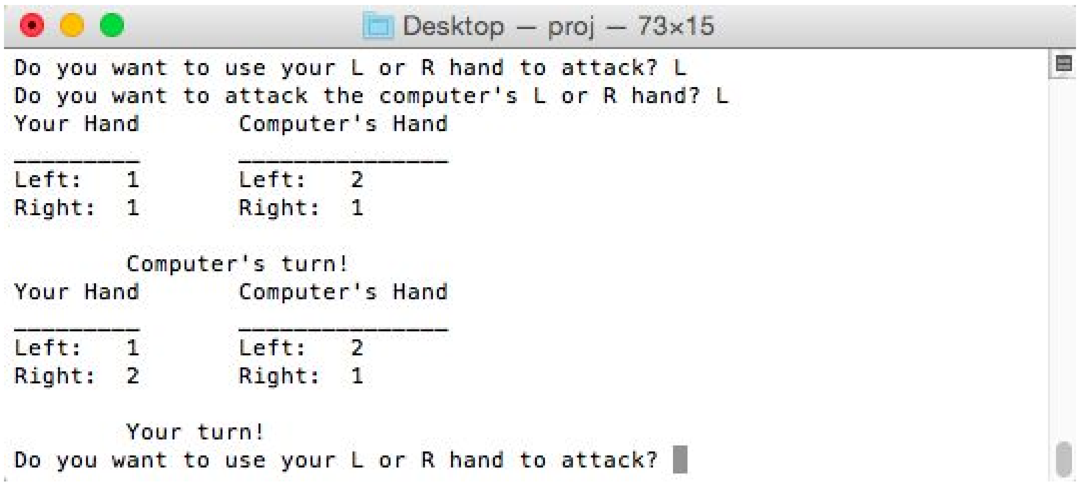
```
Pick your option: 2
1. Both you and your opponent must hold out one finger from each hand.

2. You start first.

3. Tap one of your opponent's hands with one of your fingers. Your opponent must
 then hold out one additional finger on the hand you tapped (for a total of two)
 because the hand you used to tap them with had one finger held out.

4. Let your opponent tap one of your hands.

        - If they tap you with the hand that has one finger held out,
        you must hold out one additional finger on your hand that they tapped
        (totaling two).
        - If they tap you with their hand that has two fingers held out,
        then you must add two fingers to your hand that they tapped
        (totaling three).

5. Keep taking turns tapping hands and adding fingers, but when a hand has five
fingers held out, that is called a 'dead hand'.
Put dead hands behind the player's back. The person who reaches two dead hands f
irst loses.

1. Start Game
2. How To Play
3. Quit
Pick your option: 
```

3. Starting the Game



```
Welcome to Chopsticks!

1. Start Game
2. How To Play
3. Quit
Pick your option: 1
Your Hand        Computer's Hand
_________        _______________
Left:    1       Left:    1
Right:   1       Right:   1

        Your turn!
Do you want to use your L or R hand to attack?
```

- It is the player's turn first. Input left (L/l) or right (R/r) to pick which hand to use to attack the computer.

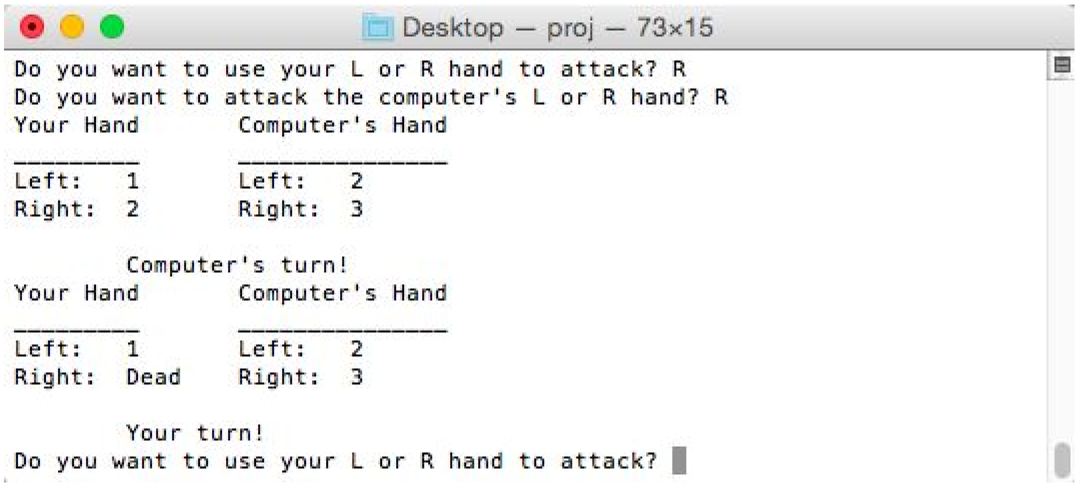


```
Do you want to use your L or R hand to attack? L
Do you want to attack the computer's L or R hand? L
Your Hand        Computer's Hand
_________        _______________
Left:    1       Left:    2
Right:   1       Right:   1

        Computer's turn!
Your Hand        Computer's Hand
_________        _______________
Left:    1       Left:    2
Right:   2       Right:   1

        Your turn!
Do you want to use your L or R hand to attack?
```

- After picking a hand to use to attack and which computer hand TO attack, the program should output an updated chart of how many fingers each person has.

4. What is “Dead”?

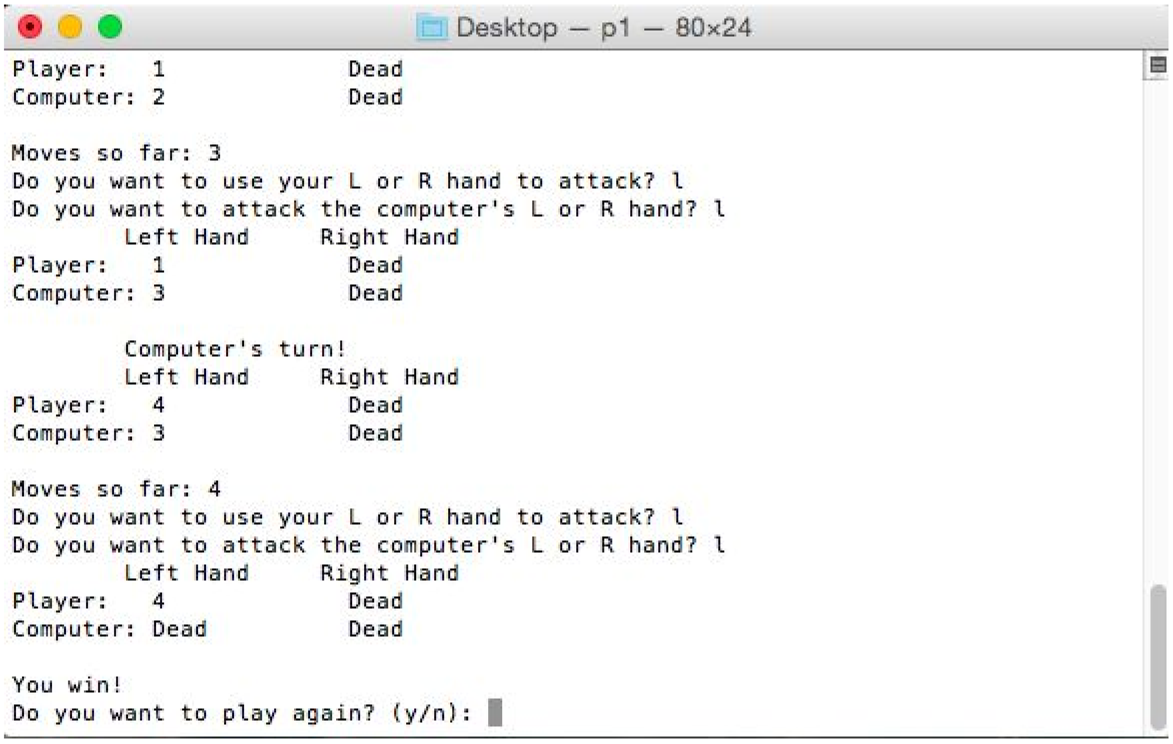
- When one hand has 5 or more fingers on it, it is considered “dead”



```
Do you want to use your L or R hand to attack? R
Do you want to attack the computer's L or R hand? R
Your Hand       Computer's Hand

_________       _______________
Left:    1      Left:    2
Right:   2      Right:   3

        Computer's turn!
Your Hand       Computer's Hand

_________       _______________
Left:    1      Left:    2
Right:   Dead   Right:   3

        Your turn!
Do you want to use your L or R hand to attack?
```

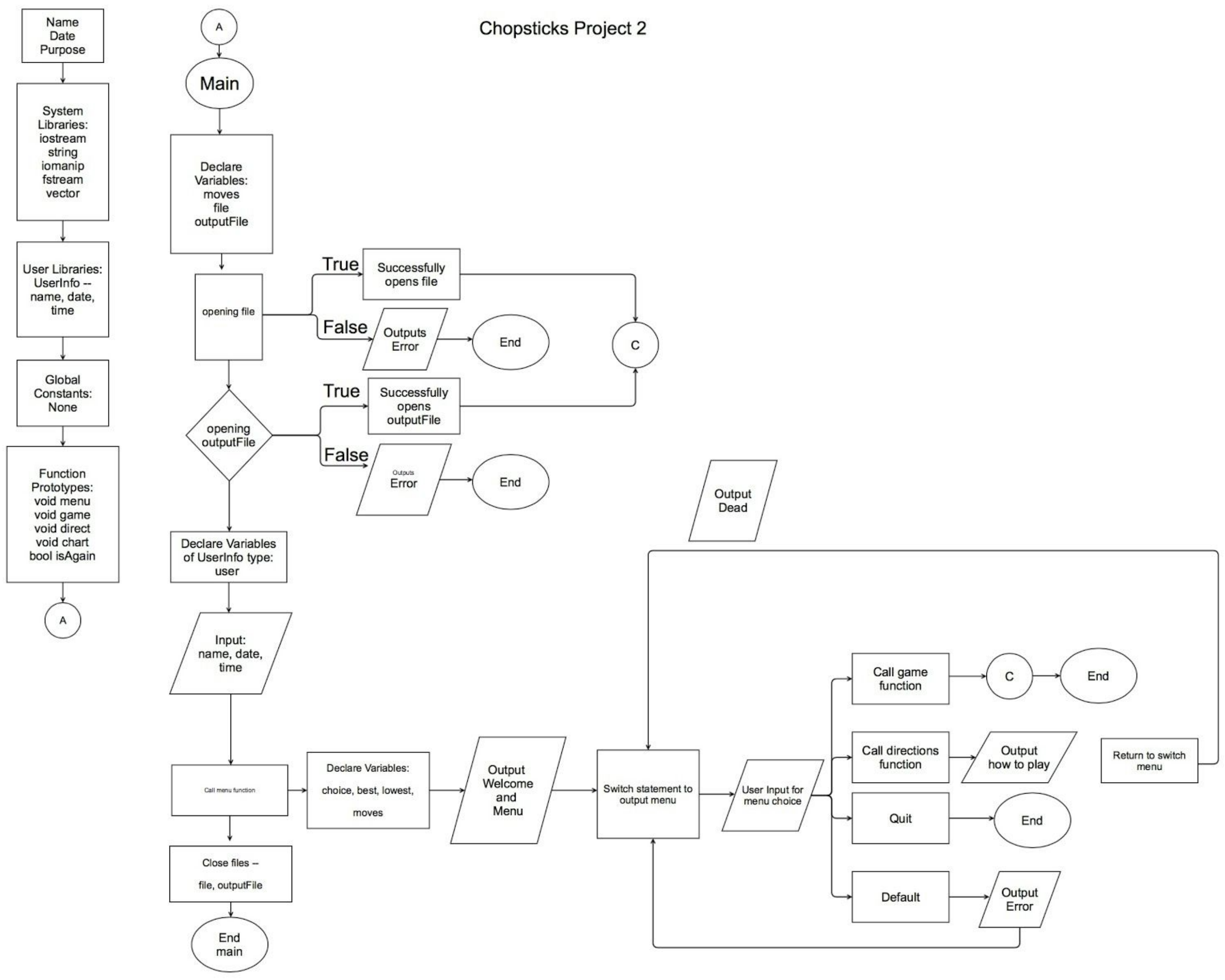
- This means that the player’s right hand is “dead” and can no longer be used to attack the computer

5. Winning!



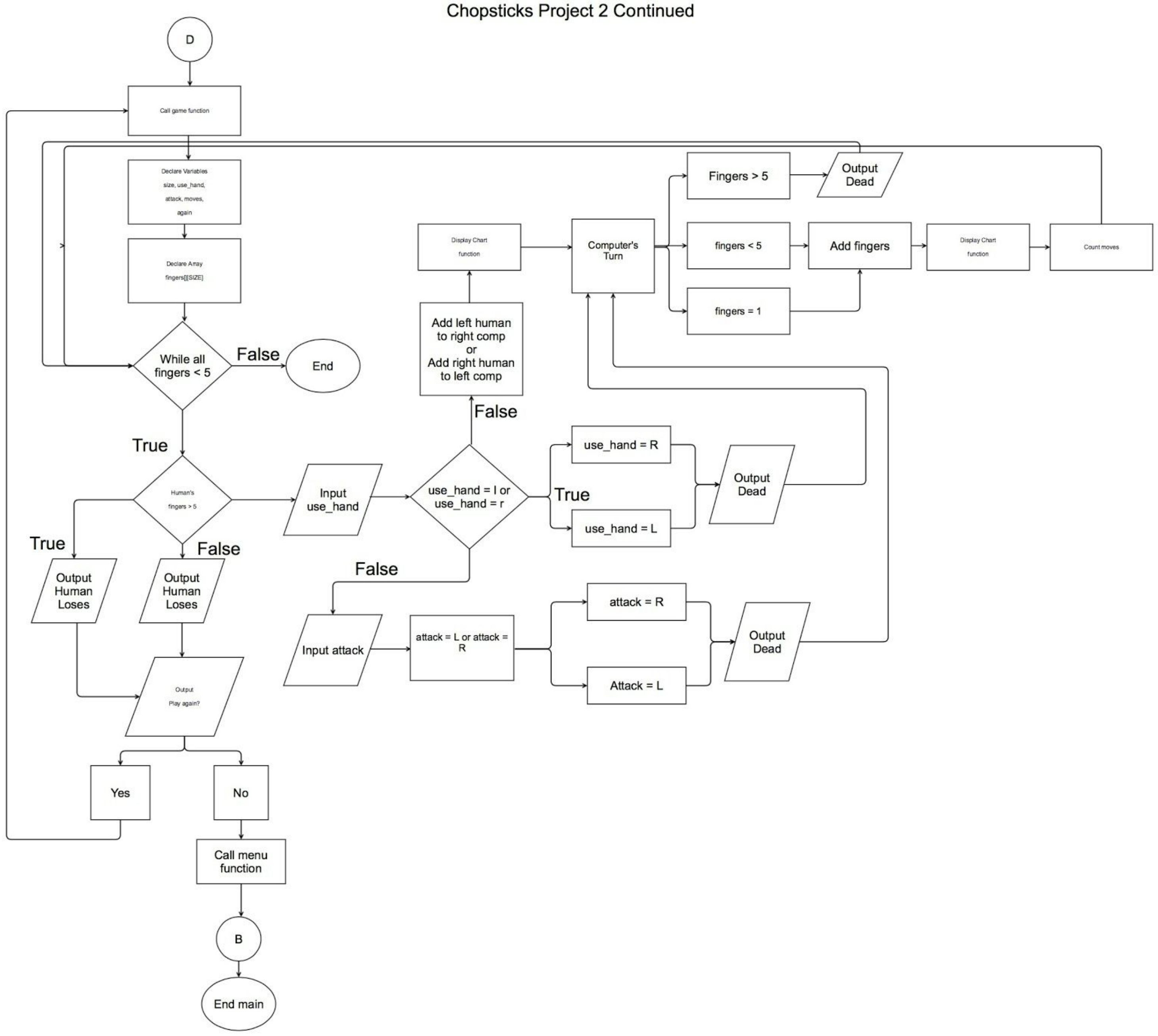
```
Player:    1            Dead
Computer: 2             Dead

Moves so far: 3
Do you want to use your L or R hand to attack? l
Do you want to attack the computer's L or R hand? l
        Left Hand     Right Hand
Player:    1            Dead
Computer: 3             Dead

        Computer's turn!
        Left Hand     Right Hand
Player:    4            Dead
Computer: 3             Dead

Moves so far: 4
Do you want to use your L or R hand to attack? l
Do you want to attack the computer's L or R hand? l
        Left Hand     Right Hand
Player:    4            Dead
Computer: Dead          Dead

You win!
Do you want to play again? (y/n):
```

Flowchart



Chopsticks Project 2
Name Date Purpose
System Libraries: iostream string iomanip fstream vector
User Libraries: UserInfo -- name, date, time
Global Constants: None
Function Prototypes: void menu void game void direct void chart bool isAgain
A
Main
Declare Variables: moves file outputFile
opening file
True
Successfully opens file
False
Outputs Error
End
C
opening outputFile
Successfully opens outputFile
Outputs Error
Declare Variables of UserInfo type: user
Input: name, date, time
Call menu function
Declare Variables: choice, best, lowest, moves
Output Welcome and Menu
Switch statement to output menu
User Input for menu choice
Call game function
Call directions function
Output how to play
Quit
Default
Output Error
Return to switch menu
Output Dead
Close files -- file, outputFile
End main

Chopsticks Project 2 Continued



Pseudo-Code

```
System libraries: iostream, string, iomanip, fstream
    -   string name, date, time
User libraries: UserInfo
Global constants: BEST
Function prototypes:
        void game(ofstream &outputFile, fstream &file);
        void direct();
        void menu(ofstream &outputFile, fstream &file);
        void chart(short fingers[][2]); bool isAgain();
I/O stream namespace

Main
    -   Declare Variables: moves, fstream file, ofstream outputFile
    -   Open file and outputFile
    -   Output error if files cannot open
    -   Menu
    -   Ask for user Info to put into UserInfo structure
    -   Switch Statement (choice)
            1.  Game
            2.  Directions
            3.  Quit
    -   Close file and outputFile
End main

Direction function
    -   Output directions to play
    -   End

Chart Function
    -   Declare Variables: int COMP = 2, PERS = 2;
    -   Chart of fingers
    -   Output chart

Boolean isAgain Function
    -   If user input = ‘y’ then return to game function
    -   If user input = ‘n’ then return to menu

Game function
    -   Declare variables: int SIZE; char use_hand, attack; short moves(0), again(0);
    -   Declare array: short fingers[][SIZE]
```

- *Call in chart function to display chart*
- *Set while loop to display who wins/loses if all fingers > 5*
- *Ask for user input on which hand to use*
- *User input on which hand to attack*
- *Loop to output error if user inputs a “dead” hand*
- *Computer’s turn*
- *If comp’s fingers > 5, output dead*
- *If comp’s fingers < 5 then add them to the human’s fingers*
- *Display updated “chart” function again*
- *Count the number of moves*
- *Output the number of moves so far*
- *End game function*

Major Variables

Type	**Variable Name**	**Description/Function**	**Location/Line** (1st occurrence)
char	choice	used for the menu in the switch statement	82
	use_hand	the hand that the player wants to use (L/R)	201
	attack	the hand the player uses to attack (L/R)	202
short	moves	number of moves to win	203
	best	best score up to date initialized to 5	87
	lowest	lowest score up to date	87
short vector	vector<short>moves(best);	vector array to sort through and tell what is best score	89
float	count	used to count how many moves to win	95
ofstream	outputFile	records how many moves to win	47-48
fstream	file	records all of the “wins”	37-38
bool	bool status;	tells whether or not user wants to play again	180-182
string	name	gets user name for UserInfo structure	21
	date	gets date for UserInfo structure	22
	time	gets time for UserInfo structure	23
int	comp	initializes computer’s hands	159
	pers	initializes person’s hands	160

C++ Concepts

Chapter	Concept	Syntax/Keywords	Location
2	Display Output: cout	cout <<	63

	Assigning Statements	char use_hand, attack, again;
		short moves;
		ofstream outputFile;
			201-206

	Arithmetic Operators	fingers[0][1] += fingers[1][0];	344
3	Read Input: cin	cin >> use_hand;	157
	Evaluating Mathematical Expressions	fingers[0][1] += fingers[1][0];	344-390
	Strings	string name, date, time;	21-23

4	if/else Statement	if (l_person >= 5)
		cout << "Dead" << '\t';
		else
		cout << l_person << '\t';
			127-130

	if/else-if Statement	if (fingers[1][0] >= 5 && fingers[0][1] >= 5)
		fingers[0][0] += fingers[1][1];
		
		else if (fingers[1][1] >= 5 && fingers[0][0] >= 5)
		fingers[0][1] += fingers[1][0];
			344-390

	Menu-Driven Programs	void menu(ofstream &outputFile, fstream &file) ;	80
	Switch statements	switch(choice)	101-135
5	while Loop	while (fingers[0][0] < 5 || fingers[0][1] < 5 || fingers[1][0] < 5 || fingers[1][1] < 5)	109
	for Loop	for (int row=0; row<PERS; row++)	164

6	Defining and Calling Functions	game(outputFile, file);
		direct();
		isAgain(again);
		chart(fingers);
			104, 109, 217, 482

	ofstream Output File	outputFile << "Name: " << user.name << endl;	61

7	Structures	struct UserInfo
		{
		string name; // user name
		string date; // date played
		string time; // time played
		};
			19-23

8	2-D Arrays	short fingers[][SIZE] = {{1,1}, {1,1}};	198

	Sorting Dynamic 1-D Arrays/Vectors	while (!file.eof()){
		for (int i=0; i<moves.size(); i++)
		{ file >> moves[best];
		if (best < lowest)
		moves[best] = lowest;
		else if (file.eof())
		cout << "End of file." << endl;}}
			114

	Vectors	vector<short>moves(best);	88

Program Code

```
/*
 * File:   proj.cpp
 * Author: Bryanna Phan
 * Purpose: Play chopsticks with the computer
 * PROJECT #1
 * Created on July 14, 2015, 12:14 AM
 */

// System Libraries
#include <iostream>
#include <string>
#include <iomanip>
#include <fstream>
#include <vector>

using namespace std;

// user libraries
struct UserInfo
{
	string name; // user name
	string date; // date played
	string time; // time played
};

// global constants
// function prototypes
void menu(ofstream &outputFile, fstream &file); // outputs menu and switch function
void direct(); // void function to output instructions on how to play
void game(ofstream &outputFile, fstream &file); // void function for the game
void chart(short fingers[][2]); // function for the actual game
bool isAgain();

// exeuction for main begins here:
int main() {

	short moves;
	fstream file("moves.txt", ios::in | ios::out | ios::app);

	if (!file)
	{
```

```
                cout << "Error in opening the file.";
                return 0;
        }

        // opening a file to write INTO
        ofstream outputFile;
        outputFile.open("info.txt", ios::out | ios::app);

        if (!outputFile)
        {
                cout << "Error in opening the file.";
                return 0;
        }

        // asking for user information to write to file
        UserInfo user; // user is a UserInfo structure
        cout << "Please enter the following information." << endl;
        cout << "Name: ";
        getline(cin, user.name);
        outputFile << "Name: " << user.name << endl;

        cout << "Date (ex: 07/02/12): ";
        getline(cin, user.date);
        outputFile << "Date: " << user.date << endl;

        cout << "Time (ex: 12:03 AM): ";
        getline(cin, user.time);
        outputFile << "Time: " << user.time << endl;

        menu(outputFile,file);
   cin.ignore();

file.close();
outputFile.close();
return 0;
}


void menu(ofstream &outputFile, fstream &file)
{       // Declare Variables
        char choice(0); // used for switch statement
        // int best(0); // best score
        // int lowest; // lowest score aka best score
// 1-D array and vectors
short best = 5, lowest;
vector<short>moves(best);

// introduction
cout << "Welcome to Chopsticks!" << endl << endl;

while (choice != 4)
{
	cout << "1. Start Game" << endl;
	cout << "2. How To Play" << endl;
	cout << "3. Quit" << endl;
	cout << "Pick your option: ";
	cin >> choice;;

	switch(choice) // switch statement for the menu
	{
		case '1': {
			game(outputFile, file);
			break;
		}

		case '2': {
			direct();
			break;
		}

		/* case '3': {
			while (!file.eof())
			{
				for (int i=0; i<moves.size(); i++)
				{
					file >> moves[best];
					if (best < lowest)
						moves[best] = lowest;
					//else if (file.eof())
						//cout << "End of file." << endl;
				}
			}
			cout << "Your best score is: " << moves[best] << endl;
			break; */
		}
```

```
                        case '3': {
                                cout << "The game has ended." << endl;
                                return;
                                break;
                        }
                        default: cout << "You didn't enter an option between 1-3. Please try
again." << endl;
                }
        }
}

void direct() // output directions to play
{
        cout << "1. Both you and your opponent must hold out one finger from each hand." <<
endl << endl;
        cout << "2. You start first." << endl << endl;
        cout << "3. Tap one of your opponent's hands with one of your fingers. Your opponent
must then hold out one additional finger on the hand you tapped (for a total of two) because the
hand you used to tap them with had one finger held out." << endl << endl;
        cout << "4. Let your opponent tap one of your hands." << endl << endl;
        cout << "       - If they tap you with the hand that has one finger held out," << endl;
        cout << "       you must hold out one additional finger on your hand that they tapped" <<
endl;
        cout << "       (totaling two)." << endl;
        cout << "       - If they tap you with their hand that has two fingers held out," << endl;
        cout << "       then you must add two fingers to your hand that they tapped" << endl;
        cout << "       (totaling three)." << endl << endl;
        cout << "5. Keep taking turns tapping hands and adding fingers, but when a hand has
five fingers held out, that is called a 'dead hand'." << endl;
        cout << "Put dead hands behind the player's back. The person who reaches two dead
hands first loses." << endl << endl;
}


// function to display the chart of fingers in each player's hand
void chart(short fingers[][2])
{       // declare variables
        int COMP = 2; // computer's hands
        int PERS = 2; // person's hands
        // chart to display initial fingers
        cout << '\t'  << "Left Hand     Right Hand" << endl;
        cout << "Player:   ";
```

```
	for (int row=0; row<PERS; row++)
	{
		for (int col=0; col<COMP; col++)
			{
				if (fingers[row][col] >= 5)
					cout << "Dead" << '\t' << '\t';
				else
					cout << fingers[row][col] << '\t' << '\t';
			}
		cout << endl;
		if (row < 1)
			cout << "Computer: ";
	}
cout << endl;
}

bool isAgain(char again)
{
	bool status;

	if (again == 'y')
		status = true;

	else if (again == 'n')
		status = false;
return status;
}

void game(ofstream &outputFile, fstream &file) { // function for the game

	// declare array size
	const int SIZE = 2;

	// declare 2D array
	short fingers[][SIZE] = {{1,1}, {1,1}}; // initialize fingers to one on each hand

	// variables for program
	char use_hand, // hand player uses to attack
	attack; // the opponent's hand being attacked by player
	short moves(0); // number of moves
	char again; // ask if user wants to play again or not

	chart(fingers); // calling in chart function to display chart
	while (fingers[0][0] < 5 || fingers[0][1] < 5 || fingers[1][0] < 5 || fingers[1][1] < 5)
	{

		if (fingers[0][0] >= 5 && fingers[0][1] >= 5)
		{
			cout << "You lose." << endl;
			outputFile << "You lost in: " << moves << " moves" << endl;
			cout << "Do you want to play again? (y/n): ";
			cin >> again;
			if (isAgain(again))
				game(outputFile, file);
			else if (isAgain(again))
				return; break;
		}
		else if (fingers[1][0] >= 5 && fingers[1][1] >= 5)
		{
			cout << "You win!" << endl;
			outputFile << "You won in: " << moves << " moves" << endl;
			file << moves << endl;
			cout << "Do you want to play again? (y/n): ";
			cin >> again;
			if (isAgain(again))
				game(outputFile, file);
			else if (isAgain(again))
				return; break;
		}

		// getting user input
		cout << "Do you want to use your L or R hand to attack? ";
		cin >> use_hand;
		use_hand = tolower(use_hand);

		if (use_hand == 'l' && fingers[0][0] >= 5)
		{
			while (use_hand == 'l')
			{
				cout << "That hand is dead. Please choose your right hand to
attack: ";
				cin >> use_hand;
				use_hand = tolower(use_hand);
			}
		}

		if (use_hand == 'r' && fingers[0][1] >= 5)
		{
			while (use_hand == 'r')
			{
				cout << "That hand is dead. Please choose your left hand to
attack: ";
				cin >> use_hand;
				use_hand = tolower(use_hand);
			}
		}

		cout << "Do you want to attack the computer's L or R hand? ";
		cin >> attack;
		attack = tolower(attack);

		if (attack == 'l' && fingers[1][0] >= 5)
		{
			while (attack == 'l')
			{
				cout << "That hand is already dead. Please choose to attack the
Computer's right hand: ";
				cin >> attack;
				use_hand = tolower(use_hand);
			}
		}

		else if (attack == 'r' && fingers[1][1] >= 5)
		{
			while (attack == 'r')
			{
				cout << "That hand is already dead. Please choose to attack the
Computer's left hand: ";
				cin >> attack;
				use_hand = tolower(use_hand);
			}
		}

		//human left:	[0][0]
		//human right:	[0][1]
		//comp left:	[1][0]
		//comp right:	[1][1]
```

```
if (use_hand == 'l')
{
        if (attack == 'l')
                fingers[1][0] += fingers[0][0];

        else if (attack == 'r')
                fingers[1][1] += fingers[0][0];
}

else if (use_hand == 'r')
{
        if (attack == 'l')
                fingers[1][0] += fingers[0][1];

        else if (attack == 'r')
                fingers[1][1] += fingers[0][1];
}

chart(fingers); // output updated chart of fingers

if (fingers[0][0] >= 5 && fingers[0][1] >= 5)
{
        cout << "You lose." << endl;
        outputFile << "You lost in: " << moves << " moves" << endl;
        cout << "Do you want to play again? (y/n): ";
        cin >> again;
        if (isAgain(again))
                menu(outputFile, file);
        else if (isAgain(again))
                return; break;
}
else if (fingers[1][0] >= 5 && fingers[1][1] >= 5)
{
        cout << "You win!" << endl;
        outputFile << "You won in: " << moves << " moves" << endl;
        file << moves << endl;
        cout << "Do you want to play again? (y/n): ";
        cin >> again;
        if (isAgain(again))
                game(outputFile, file);
        else if (isAgain(again))
                return; break;
```

```
		}

		// computer's moves
		cout << '\t' << "Computer's turn!" << '\t' << endl;


		//human left:	[0][0]
		//human right:	[0][1]
		//comp left:	[1][0]
		//comp right:	[1][1]

		// if 1 C-hand = dead & 1 P-hand = dead
		if (fingers[1][0] >= 5 && fingers[0][1] >= 5)
			fingers[0][0] += fingers[1][1];

		else if (fingers[1][1] >= 5 && fingers[0][0] >= 5)
			fingers[0][1] += fingers[1][0];

		else if (fingers[0][0] >= 5 && fingers[1][0] >= 5)
			fingers[0][1] += fingers[1][1];

		else if (fingers[0][1] >= 5 && fingers[1][1] >= 5)
			fingers[0][0] += fingers[1][0];

		// if only one hand is dead
		else if (fingers[0][0] >= 5 && fingers[1][1] < 5 && fingers[1][0] < 5 && fingers[0][1]
< 5 )
		{
			if (fingers[1][1] > fingers[1][0])
				fingers[0][1] += fingers[1][1];

			else if (fingers[1][0] >= fingers[1][1])
				fingers[0][1] += fingers[1][0];
		}

		else if (fingers[0][1] >= 5 && fingers[1][1] < 5 && fingers[1][0] < 5 && fingers[0][0]
< 5)
		{
			if (fingers[1][1] > fingers[1][0])
				fingers[0][0] += fingers[1][1];

			else if (fingers[1][0] >= fingers[1][1])
				fingers[0][0] += fingers[1][0];
		}

		else if (fingers[1][0] >= 5 && fingers[0][1] < 5 && fingers[0][0] < 5 && fingers[1][1]
< 5)
		{
			if (fingers[0][0] > fingers[0][1])
				fingers[0][0] += fingers[1][1];

			else if (fingers[0][1] >= fingers[0][0])
				fingers[0][1] += fingers[1][1];
		}

		else if (fingers[1][1] >= 5 && fingers[0][1] < 5 && fingers[0][0] < 5 && fingers[1][0]
< 5)
		{
			if (fingers[0][0] > fingers[0][1])
				fingers[0][0] += fingers[1][0];

			else if (fingers[0][1] >= fingers[0][0])
				fingers[0][1] += fingers[1][0];
		}


		// computer's first move
		else if (fingers[0][0] == 1 && fingers[0][1] == 1 && fingers[1][0] == 1)
		{
			if (fingers[1][0] > fingers[1][1])
				fingers[0][0] += fingers[1][1];

			else if (fingers[1][1] > fingers[1][0])
				fingers[0][0] += fingers[1][0];
		}

		else if (fingers[0][0] == 1 && fingers[0][1] == 1 && fingers[1][1] == 1)
		{
			if (fingers[1][0] > fingers[1][1])
				fingers[0][1] += fingers[1][1];

			else if (fingers[1][1] > fingers[1][0])
				fingers[0][1] += fingers[1][0];
		}

		// if all fingers are in play
```

```
        else if (fingers[0][0] < 5 && fingers[0][1] < 5 && fingers[1][0] < 5 && fingers[1][1] <
5)
        {
            if (fingers[0][0] >= 2 && fingers[0][1] < 2)
            {
                if (fingers[1][0] >= 2)
                    fingers[0][0] += fingers[1][0];

                else if (fingers[1][1] >= 2)
                    fingers[0][0] += fingers[1][1];
            }
            else if (fingers[0][1] >= 2 && fingers[0][0] < 2)
            {
                if (fingers[1][0] >= 2)
                    fingers[0][1] += fingers[1][0];

                else if (fingers[1][1] >= 2)
                    fingers[0][1] += fingers[1][1];
            }

            else if (fingers[0][0] == 1 && fingers[0][1] != 1)
            {
                if (fingers[1][0] >= 4)
                {
                    if (fingers[0][1] > fingers[0][0])
                        fingers[0][1] += fingers[1][0];

                    else if (fingers[0][0] > fingers[0][1])
                        fingers[0][0] += fingers[1][0];
                }

                else if (fingers[1][1] >= 4)
                {
                    if (fingers[0][1] > fingers[0][0])
                        fingers[0][1] += fingers[1][1];

                    else if (fingers[0][0] > fingers[0][1])
                        fingers[0][0] += fingers[1][1];
                }

                else if (fingers[1][0] < 4 && fingers[1][1] < 4)
                {
                    if (fingers[1][0] > fingers[1][1])
```

```
                        fingers[0][0] += fingers[1][0];

                    else if (fingers[1][1] > fingers[1][0])
                        fingers[0][0] += fingers[1][1];
                }
            }

            else if (fingers[0][1] == 1 && fingers[0][0] != 1)
            {
                if (fingers[1][0] >= 4)
                    fingers[0][1] += fingers[1][0];

                else if (fingers[1][1] >= 4)
                    fingers[0][1] += fingers[1][1];

                else if (fingers[1][0] < 4 && fingers[1][1] < 4)
                {
                    if (fingers[1][0] > fingers[1][1])
                        fingers[0][1] += fingers[1][0];

                    else if (fingers[1][1] > fingers[1][0])
                        fingers[0][1] += fingers[1][1];
                }
            }
        }

        chart(fingers);
        moves++;
        cout << "Moves so far: " << moves;
        cout << endl;
    }
}
```