Project 2

<Master Mind>

CSC5-45276

Name: Javier Borja

Date: 7/28/2016

Table of Contents

Introduction

Master Mind

For Project 2 in CSC-5, I have decided to continue my Project 1 of Master Mind because my original program felt incomplete and improvable. I originally decided to recreate this board game because it was one of my favorite games growing up.

The basis of the game is for one player to create a code (code-maker) and for one player to break the code (code-breaker). The original board game sports 6 colored pegs and 4 holes. The code-maker must choose 4 pegs, duplicates allowed, and place them in a hidden row. This is the only concern for the code-breaker, he is allowed 8-12 turns to solve the puzzle. The code-maker gives feedback after every guess by filling 4 smaller holes with black and white pegs. The black peg corresponds to a correct colored peg and position, while a white colored peg corresponds to just a correct colored peg. The game is over when the code-breaker runs out of rows (or turns) and fails to guess the code-maker's combination or if the code-breaker successfully solves the code.

The original board game is based on an earlier pencil and paper game called Bulls and Cows, which this program resembles the most, because of the limitations of my programming skills. This game uses numbers instead of colored pegs, but the goal and general impression are the same. The user must guess the Ai's combination in 8 or 12 turns, and duplicate numbers are allowed depending on which mode the user has chosen. My version of the classic game uses 8 numbers instead of the usual 6, which means there are 8^4 or 4096 possible combinations, with duplicates numbers allowed.

Tutorial

This program is straightforward and easy to understand. Your experience begins with a welcome screen, in here you will input your first and last name, and then you'll be displayed the menu with 4 different options.

```
Welcome to MasterMind: A logic codebreaker game.
Enter your first and last name and press return. Omit any middle initial you might have.
Joe Sample
Welcome Joe Sample

Press 1 to play Normal Mode MasterMind.
Press 2 to play Hard Mode MasterMind.
Press 3 to learn how to play.
Press 4 to see the leaderboard.
Press anything else to exit.
```

These options are self-explanatory. Selecting 1 or 2 lets you play Master Mind in different difficulty settings, selecting 3 shows you how to play the game, and selecting 4 lets you see the current leaderboard. You are asked for a name in the beginning of the program so you are able to show up in the leaderboard. The leaderboard is sorted by amount of wins per session.

Leaderboard:

```
Sorted by amount of wins per session.

Bob Samson,
got 8 wins and 9 losses.

Mr Clean,
got 5 wins and 3 losses.

John Smith,
got 4 wins and 7 losses.

Javier Borja,
got 3 wins and 0 losses.

Jane Doe,
got 1 wins and 2 losses.

John Miller,
got 0 wins and 0 losses.

Lisa Anderson,
got 0 wins and 2 losses.

Can you beat Bob Samson?
```

Learn how to play:

```
Normal Mode:
Guess the Ai's 4 digit code in 8 turns.
First, type in your 4 digit combination. Numbers 1-8, no duplicates.
Next, the computer will tell you how you are doing.
X's = correct number and position.
O's = correct number but incorrect position.
If you don't guess a correct number you will go straight to the next turn.
If you can't guess the computer's combination in 8 turns, you lose.

Hard Mode:
The Ai may choose duplicate numbers!
But you have up to 12 turns to guess the code.
To make it worth your while, you will get 2 wins for winning.

Hints:
If you feel stuck, type in 4 zeros(0000) to get a hint.
You will get all 4 digits of the combination, least to greatest,
but you still have to guess their position.
This will not count as a win on the leaderboard.
But you can still lose, so be careful!

Good Luck!
```

Sample gameplay

```
**********************************************************************
Turn = 4
Enter a 4 digit combination using numbers 1-8, duplicates allowed
To call a hint input 4 zeros (0,0,0,0)
0000
Your hint:
2255

**********************************************************************
Turn = 5
Enter a 4 digit combination using numbers 1-8, duplicates allowed
To call a hint input 4 zeros (0,0,0,0)
2255
XXOO
**********************************************************************
Turn = 6
Enter a 4 digit combination using numbers 1-8, duplicates allowed
To call a hint input 4 zeros (0,0,0,0)
5252
XXOO
**********************************************************************
```

```
**********************************************************************
Turn = 7
Enter a 4 digit combination using numbers 1-8, duplicates allowed
To call a hint input 4 zeros (0,0,0,0)
2525
XXOO
**********************************************************************
Turn = 8
Enter a 4 digit combination using numbers 1-8, duplicates allowed
To call a hint input 4 zeros (0,0,0,0)
5225
OOOO
**********************************************************************
Turn = 9
Enter a 4 digit combination using numbers 1-8, duplicates allowed
To call a hint input 4 zeros (0,0,0,0)
2552
XXXX
**********************************************************************
You used a hint, win doesn't count.
```

Project Summary

Project size	**456 Lines**
Lines of code	398 Lines
Comment lines	38 Lines
Blank lines	20 Lines

Project 1 was developed in versions 1-3; Project 2 was continued in version 4. I will be summarizing the development cycles in brief synopses. You may see the development process in my GitHub repository. Running code from previous versions may not work and is not advised.

https://github.com/javierborja95/JB_CSC5/tree/master/Project

Version 1

Most of the program was written in version one. The main game and turn functions were developed in this stage. Reading the rules from another file and writing to another file was not working properly and many debugging output statements are included.

Version 2

Two new functions were added. A menu function which displays the menu and the results function that displays the results of the game. Formatting was improved in the game function and the rules function was working properly.

Version 3

Output formatting is finalized in this version. Win/loss counter is working correctly, writing results to file working properly, and all debugging statements are removed.

Version 4

Project 2 begins in this stage, major variables are removed and replaced with arrays.

Version 5

Created 2 new functions, Normal mode and Hard mode of the Master Mind game. Normal mode max turns were lowered to 8 turns and Hard mode was upped to 12. The turn function broke because the original game was programmed with no duplicate numbers in the code allowed.

Version 6

A help function was developed to give the player a hint if he was stuck. This was made as a compliment to the new Hard mode. The rules were updated to reflect the new modes and the results were updated to accommodate all the new options. New outputs were made depending on the mode and if a hint was used.

Version 7

Merged the Hard mode and Normal mode into a single game function. Attempts to fix the turn function were made. Further testing revealed the problem and solution.

Version 8

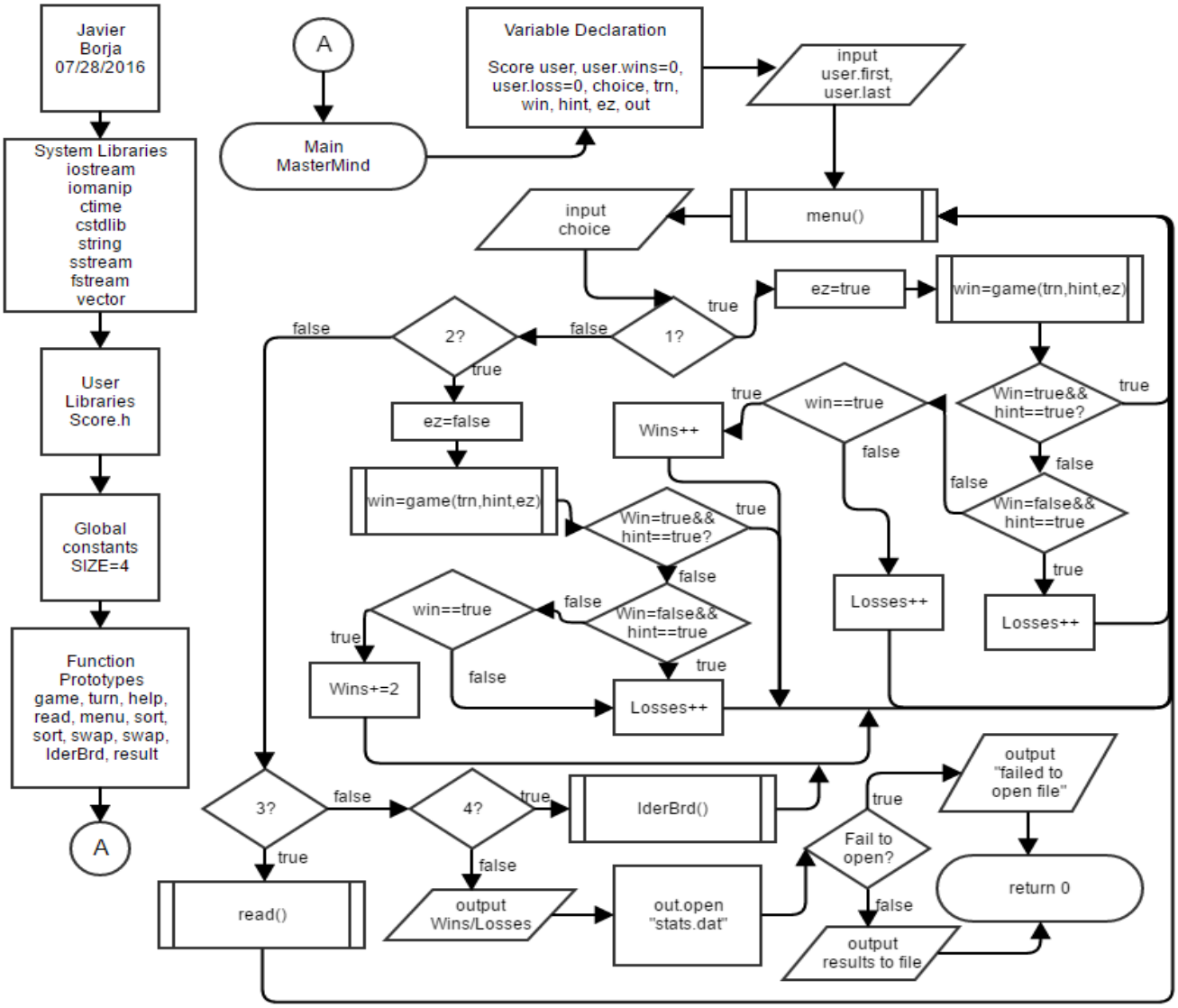
Final version. Turn function fixed for good and major improvements are made to the rest of the program. I developed a working leaderboard and created a structure to use relevant major variables in the main and leaderboard functions. Concluding output is finalized and key testing was performed to confirm a working program.

Pseudocode and Flowcharts

//Function prototypes
Game
Turn
Help
Read
Menu
Sort //Sort for hint
Sort //Sort for leaderboard
Swap
Leader Board
Result

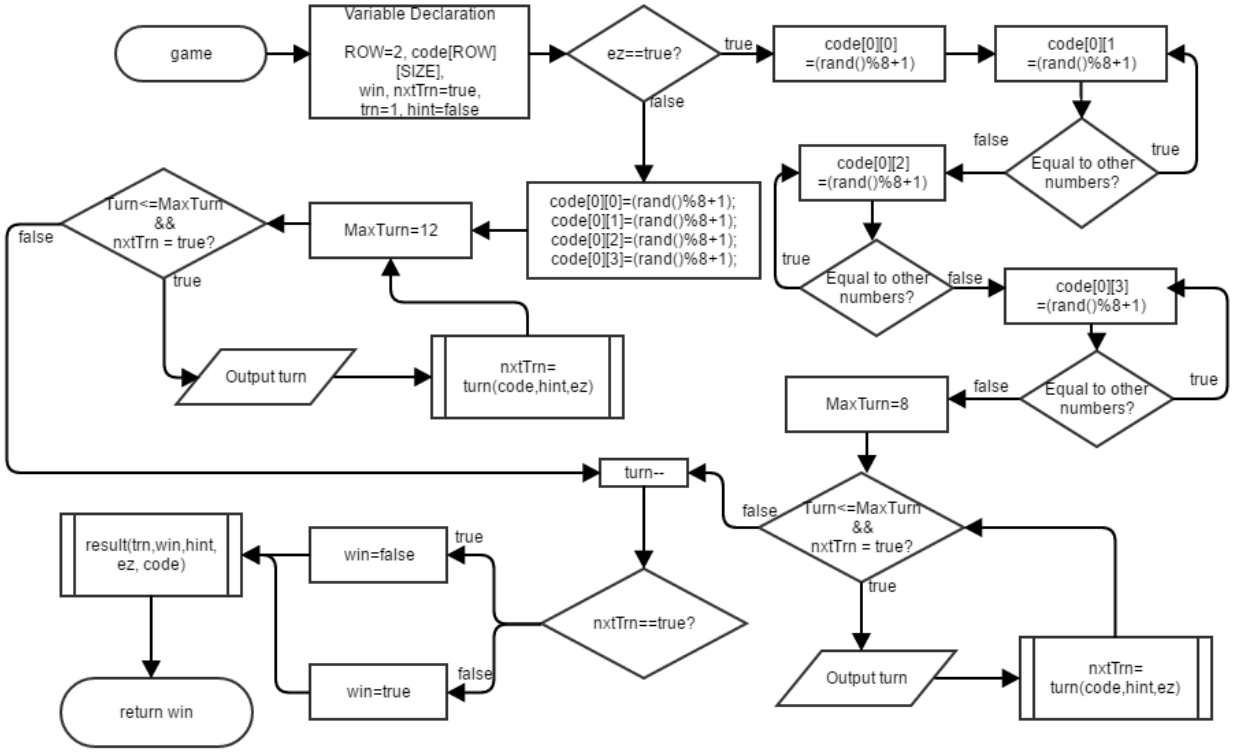
Main:
Input name
Output menu
Input choice

Switch (choice)
    Case 1: Normal Mode
    Case 2: Hard Mode
    Case 3: How to play
    Case 4: Leaderboard
    Default end program
If win
    Win++
Else
    Loss--
Output wins/losses
Write results to file



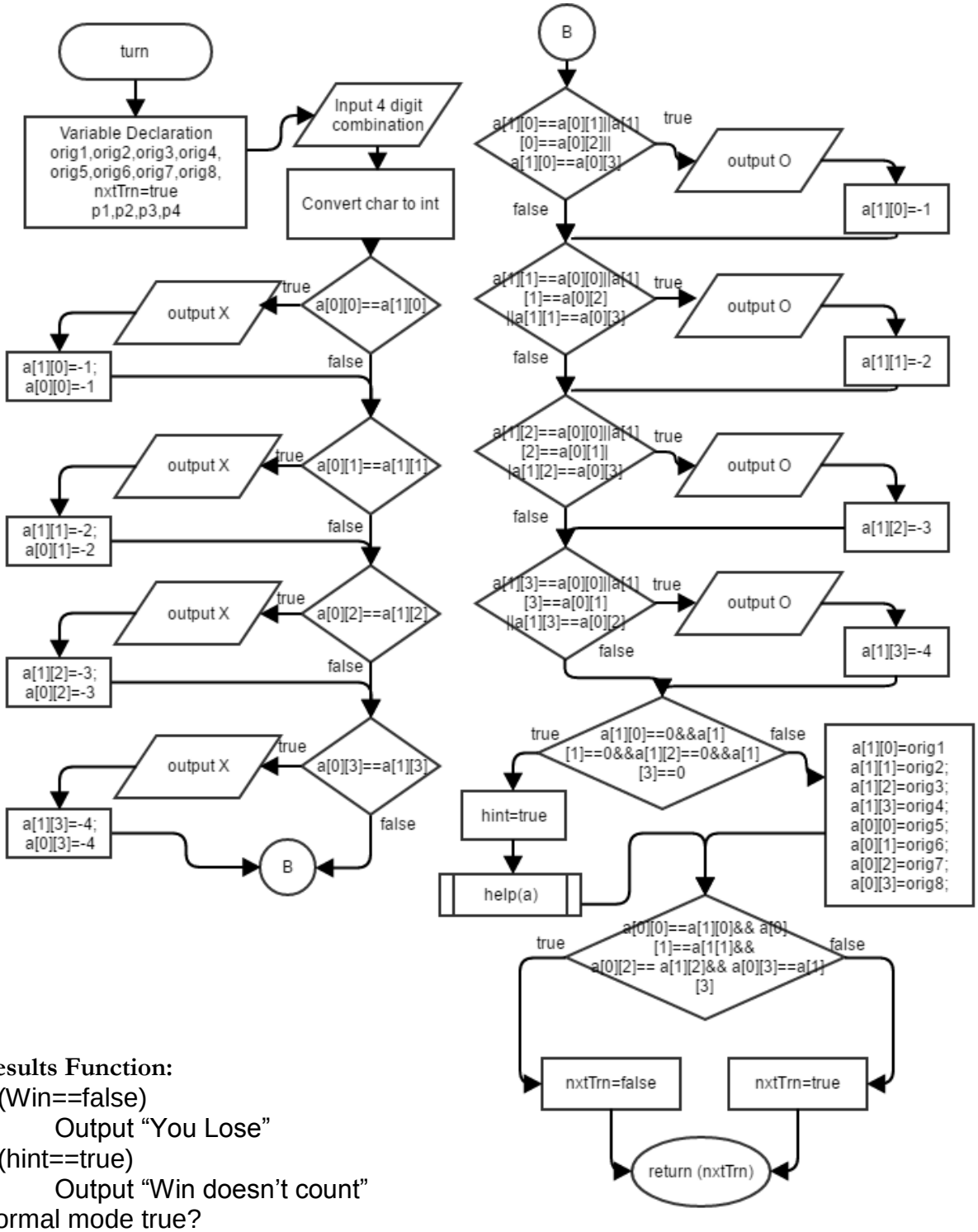
Game Function:

Get random 4-digit combination
If normal mode {
    Repeats not allowed.
    Max turns = 8
} Else {
    Repeats allowed
    Max turns = 12
}
Turn function until max turns or player wins
Results function



Turn Function:

Input player 4-digit combination
Correct number and position?
    Output X
Correct number and incorrect position?
    Output O
If 0000
    Call help function
If combinations match
    Win=true



Results Function:

If (Win==false)

Output “You Lose”

If (hint==true)

Output “Win doesn’t count”

Normal mode true?

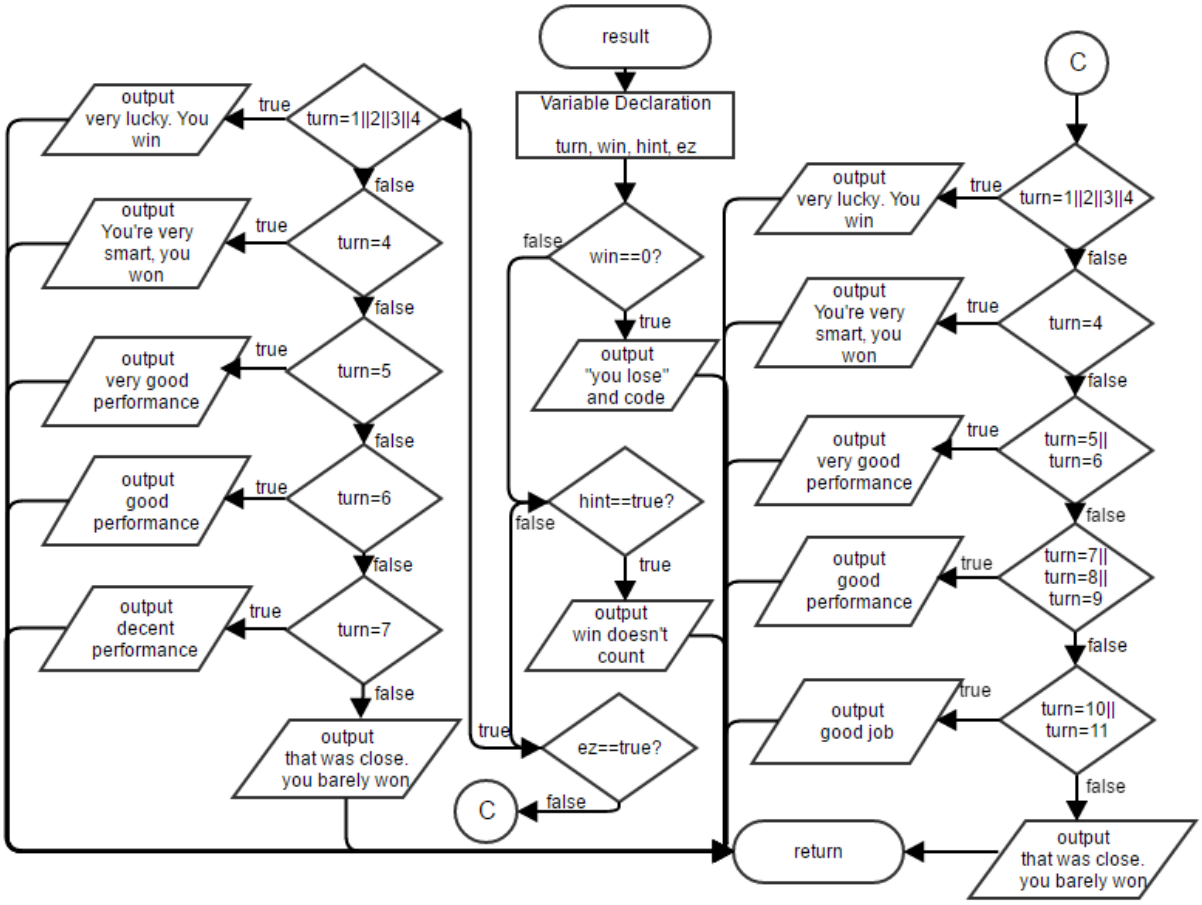
Switch(turns)

Case 1-8: Output “You win normal mode”

Else Switch(turns)

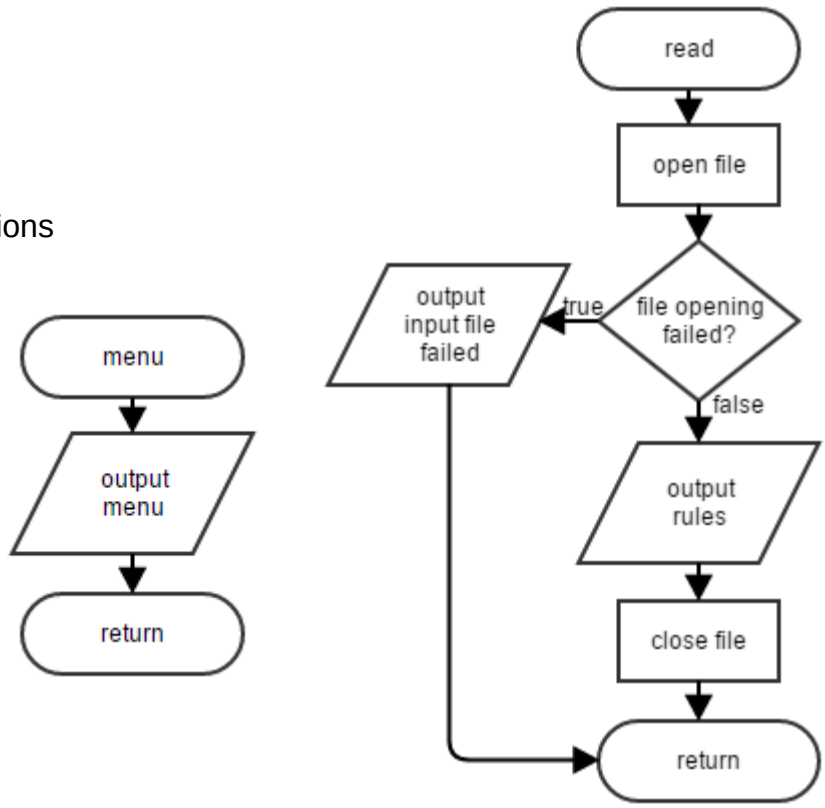
Case 1-12 Output” You win hard mode”

//Different messages display depending on which turn you win



Read Function:
Open rules file
Output rules
Close rules file

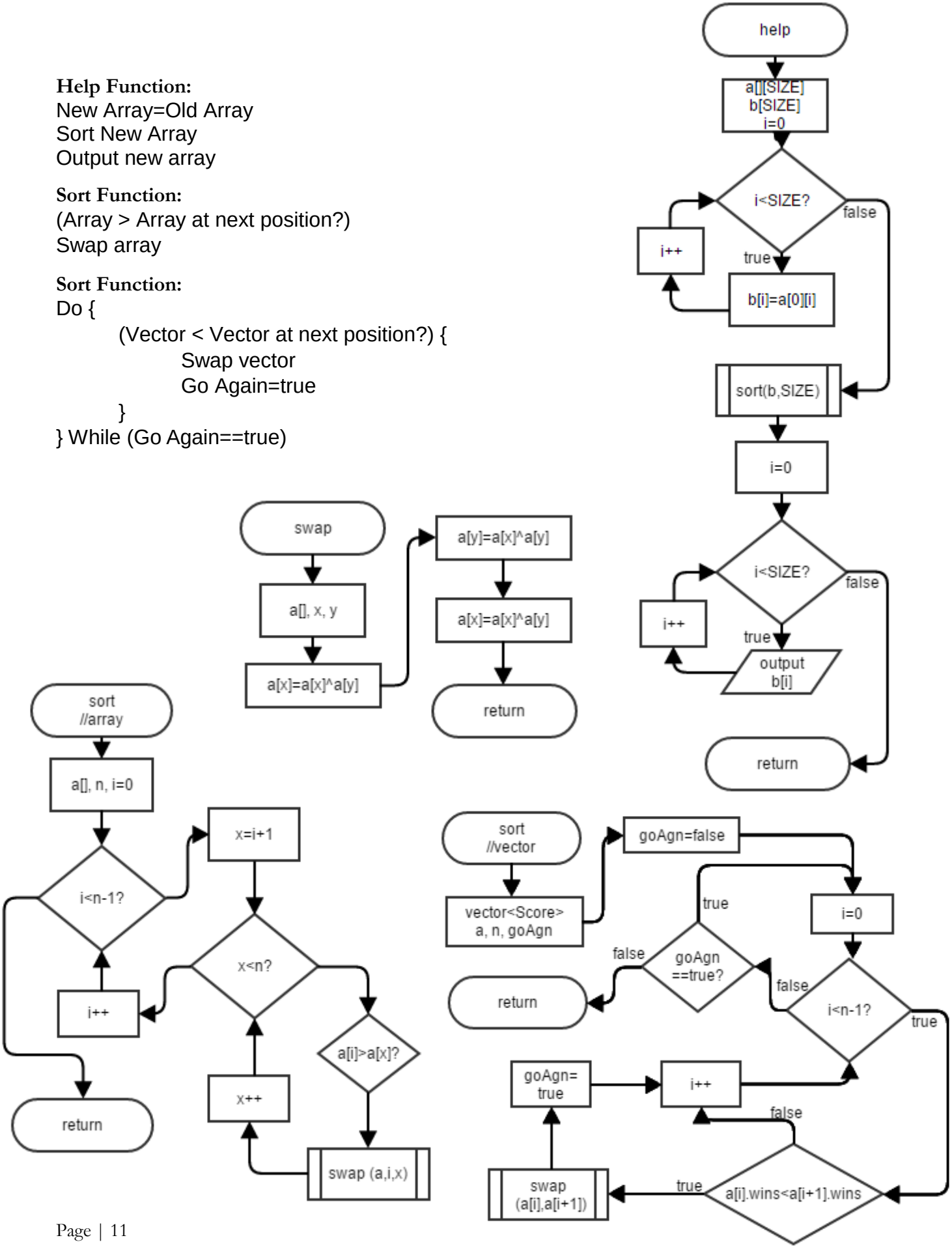
Menu Function:
Output menu options



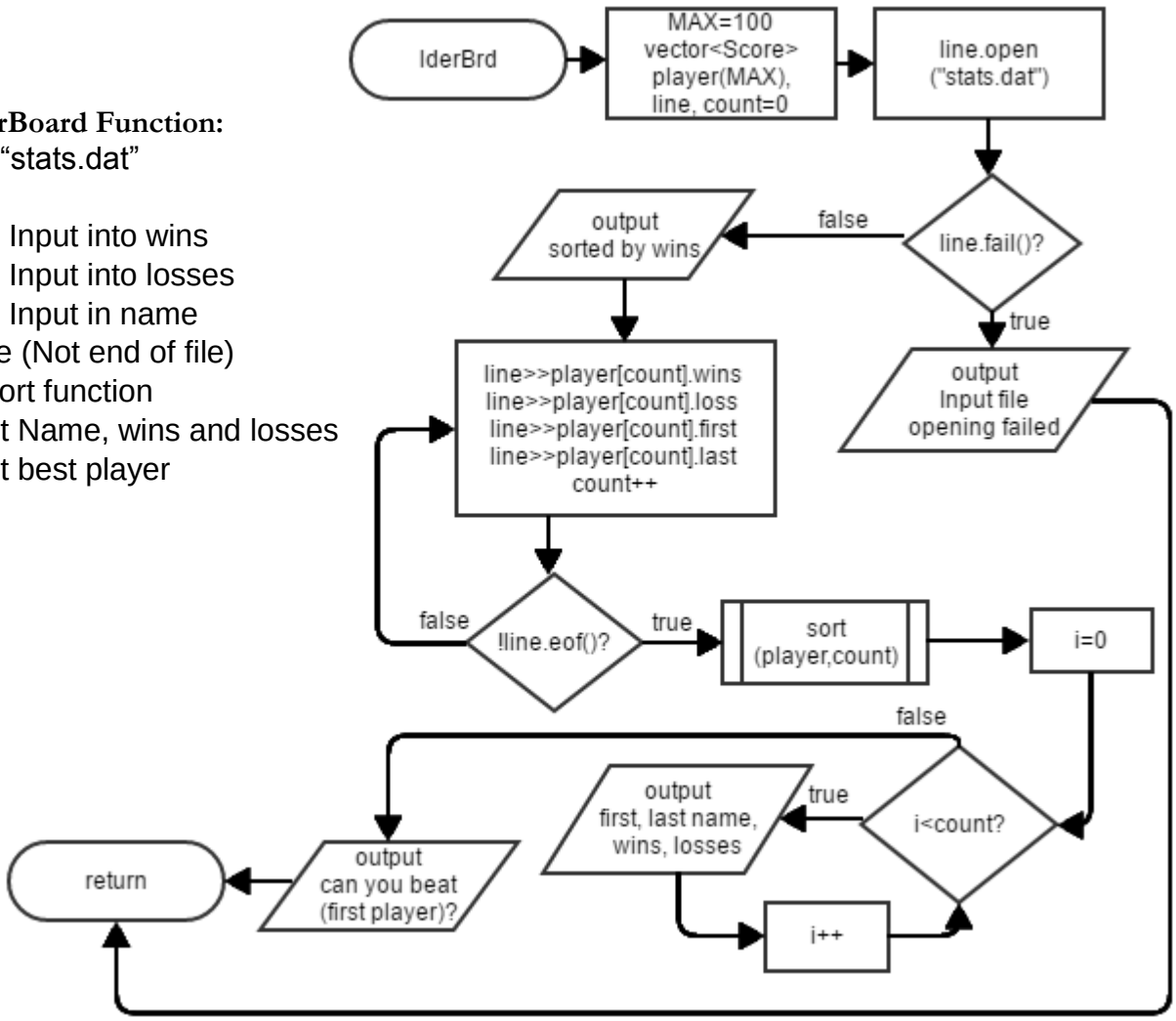
Help Function:
New Array=Old Array
Sort New Array
Output new array

Sort Function:
(Array > Array at next position?)
Swap array

Sort Function:
Do {
(Vector < Vector at next position?) {
Swap vector
Go Again=true
}
} While (Go Again==true)



LeaderBoard Function:
Open “stats.dat”
}
Input into wins
Input into losses
Input in name
} While (Not end of file)
Call Sort function
Output Name, wins and losses
Output best player



Major Variables:

Type	Variable Name	Description	Location
struct	Score	Contains user and players info	Score.h, main, lderBrd
Score	user	Contains the user’s information	main
vector<Score>	player	Contains other players’ information	lderBrd
char	choice	Menu choice	main
	p1	Player first number choice	turn
	p2	Player second number choice	turn
	p3	Player third number choice	turn
	p4	Player fourth number choice	turn
int	SIZE	Size of the code (4)	main, game, turn, result, help

	ROW	Size 2 of the code array	game
	MAX	Max amount of players in leaderboard (100)	lderBrd
	trn	What turn user is on	main, game, turn, result,
	maxTrn	Max turn	game
	orig1,orig2,orig3, orig4,orig5,orig6, orig7,orig8	Original values of code array	turn
	count	Amount of players in leaderboard	lderBrd
	code	Array of Ai code and playerCode	main, game, turn, help
	user.win	User amount of wins	main
	user.loss	User amount of losses	main
	player.wins	Players' wins	lderBrd
	player.loss	Players' losses	lderBrd
bool	win	Game result	main, game, result
	hint	Used to track if user used a hint	main, game, result
	ez	Normal mode or hard mode	main, game, result
	nxtTrn	Decides whether to go to a next turn	game, turn
	goAgn	Decides whether to loop sort function again	sort
ofstream	out	Writes results to a file	main
ifstream	line	Gets lines from files	read, lderBrd
string	user.first	User first name	main
	user.last	User last name	main
	player.first	Player's first names	lderBrd
	player.last	Player's last names	lderBrd
stringstream	ss1,ss2,ss3,ss4	Used to convert char to int	turn

Concepts utilized:

From Walter Savitch, *Problem Solving with C++*, Ninth Edition.

Chapter	Construct	Location
2.1	Variables and Identifiers	char choice;
	Assigned statements	ez=true;
2.2	Input and Output	cin>>choice; cout<<endl;
	Escape sequences	cout<<"Press 3 to learn how to play.\n"
2.3	Number types	int trn;
	Type char	char choice;
	Type bool	bool win;

	Class string	string first
2.4	If-else statement	if(win==true) user.wins+=2; else user.loss++;
	Boolean expressions	if(nxtTrn==true)

	Compound statements	if(a[0][0]==a[1][0]){
		cout<<"X"; a[1][0]=-1;
		a[0][0]=-1; }

	While loop	while(goAgn==true)
	Increment and decrement operator	user.wins++; trn--;
2.5	Naming constants	const int SIZE=4;
3.2	Nested statements	else{ switch(turn){
	Switch statements and menus	switch(choice){
3.3	For statement	for(int maxTrn=8; (trn<=maxTrn&&nxtTrn==true);trn++)
3.4	Sentinel values	"Press anything else to exit."<<endl; cin>>choice;
4.1	Functions	bool game(int &turn,bool&,bool);
4.2	Random Number Generation	code[0][0]=(rand()%8+1)
	Type Casting	srand(static_cast<unsigned int>(time(0)));
4.3	Programmer defined functions	bool turn(int[][SIZE],bool&,bool);
	Pass-by-value	result(trn,win,hint,ez,code);
	Return statements	return(nxtTrn);
4.5	Local variables	void read(){ ifstream line;
4.6	Overloading Function Names	void sort(int[],int; void sort(vector<Score>&,int);
5.1	Void Function	void menu();
5.2	Call-by-reference	bool game(int &turn,bool&,bool);
5.3	Function calling functions	void lderBrd(){ sort(player,count);
6.1	Streams and File I/O	ofstream out; ifstream line;
	Writing to File	out<<'\r'<<user.wins<<' '<<user.loss
	Appending to a File	out.open("stats.dat",ios::app);
6.2	Manipulators	cout<<"Wins this session"<<setw(5)<<"= " <<user.wins<<endl;
6.3	Defaulted Parameters	bool game(int &turn,bool&,bool=true);
7.1	Arrays	int code[ROW][SIZE];
7.2	Arrays in Functions	void help(int a[][SIZE]){
7.3	Sorting an Array	void sort(int a[],int n){
7.4	Multidimensional Arrays	int code[ROW][SIZE];
8.2	I/O with String Class	while(getline(line,string)){
8.3	Vectors	vector<Score> player(MAX);

10.1	Structures	struct Score{
		int wins; int loss; string first; string last;
		};

Other	Stringstream	ss1<<p1 ss1>>a[1][0];

Code:

```
/*
 * File:   Score.h
 * Author: Javier B
 * Created on July 27, 2016, 5:24 PM
 */
#ifndef SCORE_H
#define SCORE_H

using namespace std;

struct Score{
    int wins;
    int loss;
    string first; //Name
    string last;  //Name
};

#endif // SCORE_H

/*
 * File:   main.cpp
 * Author: Javier Borja
 * Created on July 27, 2016, 3:40 PM
 * Purpose: Player guesses a 4 digit combination against the computer.
 */

//System Libraries
#include <iostream>   //Input/ Output Stream Library
#include <iomanip>    //Output Manipulation
#include <ctime>      //Computer Time for seed
#include <cstdlib>    //Library for random number seed
#include <string>     //String library
#include <sstream>    //String Stream Library
#include <fstream>    //File I/O
#include <vector>     //Vectors

using namespace std;  //Namespace of the System Libraries

//User Libraries
```

```
#include "Score.h"

//Global Constants
const int SIZE=4; //Size of code

//Function Prototypes
bool game(int &turn,bool&,bool=true);//A single game of MasterMind
bool turn(int[][SIZE],bool&,bool);   //A single turn of mastermind
void help(int[][SIZE]);              //Hint
void read();                        //Read rules
void menu();                         //Menu
void sort(int[],int);               //Sort hint
void sort(vector<Score>&,int);       //Sort leaderboard
void swap(int[],int,int);            //Swap
void lderBrd();                     //Leaderboard
void result(int,bool,bool,bool,int[][SIZE]); //Result of game

//Execution
int main(int argc, char** argv) {
   //Set Random seed
   srand(static_cast<unsigned int>(time(0)));

   //Variables
   Score user;          //Player
   user.wins=0;         //Wins
   user.loss=0;         //Losses
   char choice;         //Menu choice
   int trn;            //Turn
   bool win;            //Game result. True results in win
   bool hint;          //Hint
   bool ez;            //Difficulty easy or hard
   ofstream out;        //Output results to file

   //Input Data
   cout<<"Welcome to MasterMind: A logic codebreaker game.\n"
         "Enter your first and last name and press return. Omit any middle"
         " initial you might have."<<endl;
   cin>>user.first>>user.last;

   //Process Data & Menu
   cout<<"Welcome "<<user.first<<" "<<user.last<<endl<<endl;
   do{
      menu();
      cin>>choice;
      cout<<endl;
      switch(choice){
```

```
            case'1':
               win=game(trn,hint);
            if(hint==true&&win==true)
               break;
            if(hint==true&&win==false){
               user.loss++;
               break;
            }
            if(win==true)
               user.wins++;
            else
               user.loss++;
            break;
            case'2':
               ez=false;
               win=game(trn,hint,ez);
            if(hint==true&&win==true)
               break;
            if(hint==true&&win==false){
               user.loss++;
               break;
            }
            if(win==true)
               user.wins+=2;
            else
               user.loss++;
            break;
            case'3':read();break;
            case'4':lderBrd();
         }
      }while (choice=='1'||choice=='2'||choice=='3'||choice=='4');

      //Output Data and output to file
      cout<<"Thanks for playing "<<user.first<<" "<<user.last<<endl;
      cout<<"Wins this session"<<setw(5)<<"= "<<user.wins<<endl;
      cout<<"Losses this session = "<<user.loss<<endl;
      out.open("stats.dat",ios::app); //Append to file
      if (out.fail())
         cout<<"Input file 1 opening failed.\n";
      out<<'\r'<<user.wins<<' '<<user.loss<<' '
              <<user.first<<' '<<user.last;
      out.close();
      return 0;
}

bool game(int &trn,bool &hint,bool ez){
```

```
//Set variables
const int ROW=2;     //Two codes, Ai choice and player choice
int code[ROW][SIZE]; //2x4 Array
bool win;
bool nxtTrn;
     nxtTrn=true;    //Decides to go to next turn

//Process Data and get random combination
trn=1;
hint=false;
if(ez==true){
   //Normal Mode
   code[0][0]=(rand()%8+1);
   do{
      code[0][1]=(rand()%8+1);
   }while (code[0][1]==code[0][0]);                    //No duplicates
   do{
      code[0][2]=(rand()%8+1);
   }while (code[0][2]==code[0][1]||code[0][2]==code[0][0]); //No duplicates
   do{
      code[0][3]=(rand()%8+1);
   }while (code[0][3]==code[0][2]||code[0][3]==code[0][1]|| //No duplicates
         code[0][3]==code[0][0]);
   for(int maxTrn=8;(trn<=maxTrn&&nxtTrn==true);trn++){//Max turns 8
      cout<<"Turn = "<<trn<<endl;
      nxtTrn=turn(code,hint,ez);
   }
}
else{
   //Hard Mode
   code[0][0]=(rand()%8+1); //Duplicates allowed
   code[0][1]=(rand()%8+1);
   code[0][2]=(rand()%8+1);
   code[0][3]=(rand()%8+1);
   for(int maxTrn=12;(trn<=maxTrn&&nxtTrn==true);trn++){//Max turns 12
      cout<<"Turn = "<<trn<<endl;
      nxtTrn=turn(code,hint,ez);
   }
}
trn--;         //Offset the extra turn from end of loop
if(nxtTrn==true)
   win=false;  //Lose if the game still wants to go to another turn
else win=true;  //Win if game doesn't need to go to another turn

//Output Result
result(trn,win,hint,ez,code);
```

```
    return(win);
}

bool turn(int a[][SIZE],bool& hint,bool ez){
    //Declare Variables
    int orig1,orig2,orig3,orig4,orig5,orig6,orig7,orig8;
    bool nxtTrn;
        nxtTrn=true;
    stringstream ss1,ss2,ss3,ss4;
    char   p1, p2, p3, p4;//Strip 4 character combination to 4 separate ints
    //Input Data
    if(ez==true)
    cout<<"Enter a 4 digit combination using numbers 1-8, no duplicates"<<endl;
    else
    cout<<"Enter a 4 digit combination using numbers 1-8, duplicates allowed\n";
    cout<<"To call a hint input 4 zeros (0,0,0,0)"<<endl;
    cin>>p1>>p2>>p3>>p4;   //Input player, 4 character combination
    //Process Data
    ss1<<p1; ss1>>a[1][0]; //Convert char to ints
    ss2<<p2; ss2>>a[1][1];
    ss3<<p3; ss3>>a[1][2];
    ss4<<p4; ss4>>a[1][3];
    orig1=a[1][0]; orig5=a[0][0]; //Copy original Values
    orig2=a[1][1]; orig6=a[0][1];
    orig3=a[1][2]; orig7=a[0][2];
    orig4=a[1][3]; orig8=a[0][3];
    //Output Data
    if(a[0][0]==a[1][0]){ //If number and position match
        cout<<"X";
        a[1][0]=-1;       //Change value to prevent duplication bugs when
        a[0][0]=-1;       //    comparing with other digits
    }                   //Prevents outputs like 'XXXOOO'
    if(a[0][1]==a[1][1]){ //If number and position match
        cout<<"X";
        a[1][1]=-2;
        a[0][1]=-2;
    }
    if(a[0][2]==a[1][2]){ //If number and position match
        cout<<"X";
        a[1][2]=-3;
        a[0][2]=-3;
    }
    if(a[0][3]==a[1][3]){ //If number and position match
        cout<<"X";
        a[1][3]=-4;
```

```
        a[0][3]=-4;
    }
    if(a[1][0]==a[0][1]||a[1][0]==a[0][2]||a[1][0]==a[0][3]){ //If numbers match
        cout<<"O";
        a[1][0]=-1;
    }
    if(a[1][1]==a[0][0]||a[1][1]==a[0][2]||a[1][1]==a[0][3]){ //If numbers match
        cout<<"O";
        a[1][1]=-2;
    }
    if(a[1][2]==a[0][0]||a[1][2]==a[0][1]||a[1][2]==a[0][3]){ //If numbers match
        cout<<"O";
        a[1][2]=-3;
    }
    if(a[1][3]==a[0][0]||a[1][3]==a[0][1]||a[1][3]==a[0][2]){ //If numbers match
        cout<<"O";
        a[1][3]=-4;
    }
    if(a[1][0]==0&&a[1][1]==0&&a[1][2]==0&&a[1][3]==0){
        hint=true;
        help(a);
    }
    a[1][0]=orig1; //Set back to original values
    a[1][1]=orig2;
    a[1][2]=orig3;
    a[1][3]=orig4;
    a[0][0]=orig5;
    a[0][1]=orig6;
    a[0][2]=orig7;
    a[0][3]=orig8;
    nxtTrn=(a[0][0]==a[1][0]&&a[0][1]==a[1][1]&&
    a[0][2]==a[1][2]&&a[0][3]==a[1][3])?false:true;
    //If combinations equal then no need for next turn.
    if(ez==true) //Different output based on mode, just for program to look pretty
    cout<<endl<<"**************************************************************";
    else
    cout<<endl<<"******************************************************************";
    cout<<endl;
    return(nxtTrn);
}

void help(int a[][SIZE]){
    //Declare Variables
    int b[SIZE]; //Original array, sorted
    //Input data
    for(int i=0;i<SIZE;i++){
```

```
        b[i]=a[0][i];
    }
    //Process Data
    sort(b,SIZE);
    //Output Data
    cout<<"Your hint:"<<endl;
    for(int i=0;i<SIZE;i++){
        cout<<b[i];
    }
    cout<<endl;
}

void sort(int a[],int n){
    for(int i=0;i<n-1;i++){
        for(int x=i+1;x<n;x++){
            if(a[i]>a[x]){
                swap(a,i,x);
            }
        }
    }
}

void swap(int a[],int x,int y){
    a[x]=a[x]^a[y];
    a[y]=a[x]^a[y];
    a[x]=a[x]^a[y];
}

void sort(vector<Score>& a,int n){
    //Declare Variables
    bool goAgn;
    //Process Data
    do{
        goAgn=false;
        for(int i=0;i<n-1;i++){
            if(a[i].wins<a[i+1].wins){
                swap(a[i],a[i+1]);
                goAgn=true;
            }
        }
    }while(goAgn==true);
}

void result(int turn, bool win,bool hint,bool ez,int a[][SIZE]){//Result of game
    //Output Data
    if(win==0){
```

```
    cout<<"You lose!\n"
        "Better luck next time.\n"
        "The code was actually:"<<endl;
    cout<<a[0][0]<<a[0][1]<<a[0][2]<<a[0][3]<<endl<<endl;
    return;
}
if (hint==true){
    cout<<"You used a hint, win doesn't count."<<endl<<endl;
    return;
}
else if(ez==true){
    switch(turn){
        case 1:
        case 2:
        case 3:
            cout<<"Wow, very lucky. You win!"<<endl;break;
        case 4:
            cout<<"Congratulations!\n"
                "You're very smart, you won in 4 turns!"
                <<endl;break;
        case 5:
            cout<<"Congratulations!\n"
                <<"You won in 5 turns. Very good performance!"
                <<endl;break;
        case 6:
            cout<<"Congratulations!\n"
                <<"You won in 6 turns. Good performance!"
                <<endl;break;
        case 7:
            cout<<"Congratulations!\n"
                <<"You won in 7 turns. Decent performance."
                <<endl;break;
        default:
            cout<<"That was close! You barely won."
                <<endl;break;
    }
}
else{
    switch(turn){
        case 1:
        case 2:
        case 3:
            cout<<"Wow, very lucky. You win!"<<endl;break;
        case 4:
            cout<<"Congratulations!\n"
                "You're very smart, you won in 4 turns!"
```

```
                <<endl;break;
            case 5:
                cout<<"Congratulations!\n"
                <<"You won in 5 turns. Very good performance!"
                <<endl;break;
            case 6:
                cout<<"Congratulations!\n"
                <<"You won in 6 turns. Very good performance!"
                <<endl;break;
            case 7:
                cout<<"Congratulations!\n"
                <<"You won in 7 turns. Good performance!"
                <<endl;break;
            case 8:
                cout<<"Congratulations!\n"
                <<"You won in 8 turns. Good performance!"
                <<endl;break;
            case 9:
                cout<<"Congratulations!\n"
                <<"You won in 9 turns. Decent performance."
                <<endl;break;
            case 10:
                cout<<"Congratulations!\n"
                <<"You won in 10 turns. Good job."
                <<endl;break;
            case 11:
                cout<<"Congratulations!\n"
                <<"You won in 11 turns. Good job."
                <<endl;break;
            default:
                cout<<"That was close! You barely won."
                <<endl;break;
        }
        cout<<"Because this is hard mode, this win counts as two wins!"<<endl;
    }
    cout<<endl;
}

void read(){ //Input Rules to display
    //Declare Variables
    ifstream line;
    string string;
    //Process and Output Data
    line.open("rules.dat");
    if(line.fail()){
        cout<<"Input file opening failed.\n";
```

```
    }
    cout<<endl;
    while(getline(line,string)){
        cout<<string<<endl;
    }
    line.close();
    cout<<endl<<endl;
}

void menu(){
    cout<<"Press 1 to play Normal Mode MasterMind.\n"
          "Press 2 to play Hard Mode MasterMind.\n"
          "Press 3 to learn how to play.\n"
          "Press 4 to see the leaderboard.\n"
          "Press anything else to exit."<<endl;
}

void lderBrd(){
    //Declare Variables
    int MAX=100;
    vector<Score> player(MAX);
    ifstream line;
    int count=0;
    //Process and Output Data
    line.open("stats.dat");
    if(line.fail()){
        cout<<"Input file opening failed.\n";
        return;
    }
    cout<<"Sorted by amount of wins per session."<<endl;
    do{
        line>>player[count].wins;
        line>>player[count].loss;
        line>>player[count].first;
        line>>player[count].last;
        count++;
    }while(!line.eof());
    sort(player,count);
    for(int i=0;i<count;i++){
        cout<<endl;
        cout<<player[i].first<<" "<<player[i].last<<",\ngot "<<player[i].wins
            <<" wins and "<<player[i].loss<<" losses."<<endl;
    }
    cout<<endl;
    cout<<"Can you beat "<<player[0].first<<" "<<player[0].last<<"?"<<endl<<endl;
}
```