Project 1

<Wheel of Fortune>

CSC17a-48096

Name: Javier Borja

Date: 10/28/2016

Table of Contents

Introduction

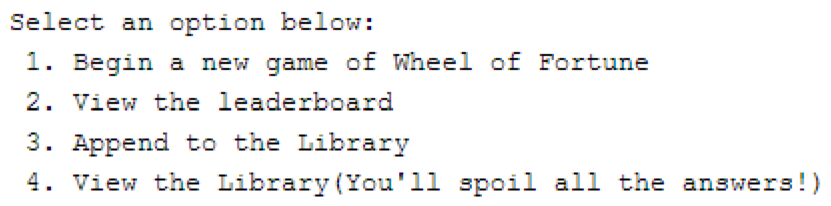
Welcome to Wheel of Fortune

For my first project this semester I decided to recreate Wheel of Fortune, with some minor changes to address many limitations. For example, this program is a single-player experience, the win/loss system is tweaked a bit and I added a point system for leaderboard tracking. But many of the classic Wheel of Fortune features are here, you can still spin the wheel, buy vowels, and guess phrases across a variety of categories.

Tutorial

(Note: You can input uppercase or lowercase characters throughout the game/menus)

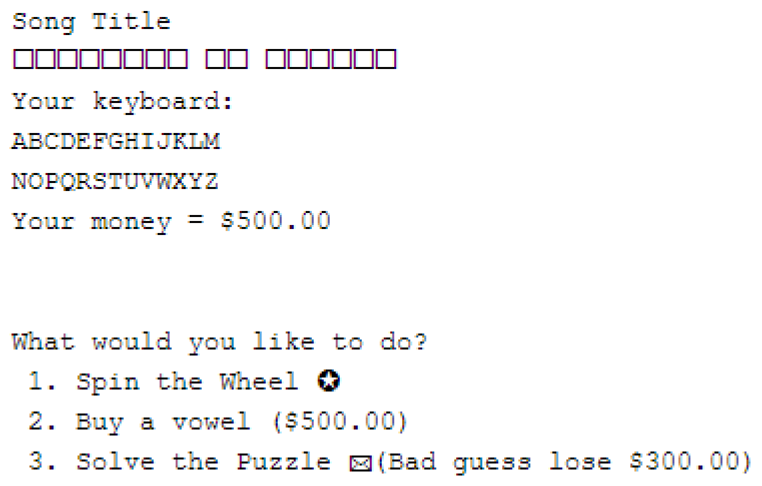
After inputting your name, you start with $500.00 and 0 Points. You will then be taken to a menu. There are four options: Play a game, view the leaderboard, add phrases to the library, or view the entire library. When you view the leaderboard, previous scores of other players will be listed from highest amount of points to the least amount of points. If you wish to append to the library, just follow the onscreen menus to do so. Again you do not need to uppercase any characters. Input validation is included in this program. Lastly, if you wish to view the library, you may do so. However, you will spoil all the answers.



```
Select an option below:
 1. Begin a new game of Wheel of Fortune
 2. View the leaderboard
 3. Append to the Library
 4. View the Library(You'll spoil all the answers!)
```

Playing a Game:

To win, you must guess the phrase; if you run out of money, you lose. Once you begin playing, you are given a category and phrase to guess. Displayed will be your hidden phrase with spaces, used/unused letters, and your money. Select an appropriate option to continue.

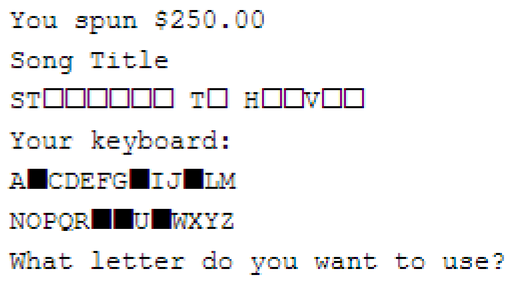


```
Song Title
□□□□□□□□ □□ □□□□□□
Your keyboard:
ABCDEFGHIJKLM
NOPQRSTUVWXYZ
Your money = $500.00


What would you like to do?
 1. Spin the Wheel ✪
 2. Buy a vowel ($500.00)
 3. Solve the Puzzle ✉(Bad guess lose $300.00)
```

Spin the Wheel:

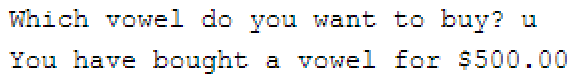
After spinning, you will be displayed a monetary value. If you correctly guess a letter, you will be awarded that amount of money and gain 10 points for each letter in the phrase that matched, else if you guess incorrectly, you will lose that amount. You can keep guessing if you have not used all the letters. Every letter you used will be blacked out and each letter you correctly guessed will be displayed.



```
You spun $250.00
Song Title
ST□□□□□□ T□ H□□V□□
Your keyboard:
A■CDEFG■IJ■LM
NOPQR■■U■WXYZ
What letter do you want to use?
```

Buy a Vowel:

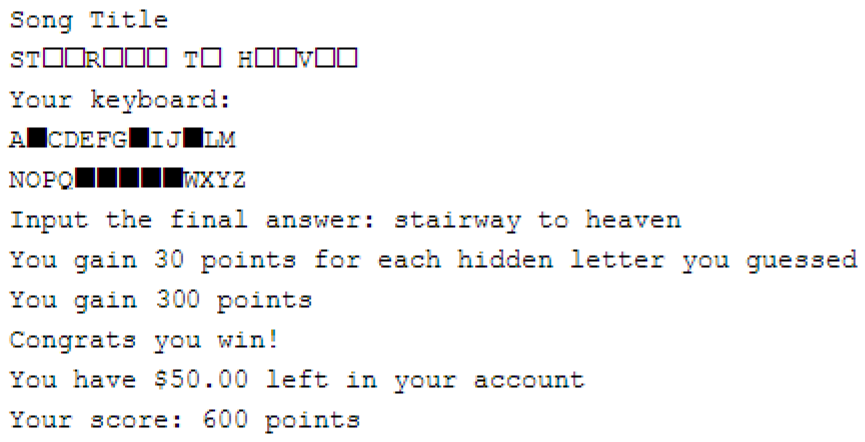
You will be displayed the same graphics as above, except you must buy a vowel. You will lose $500.00 for buying a vowel.



```
Which vowel do you want to buy? u
You have bought a vowel for $500.00
```

Solve the Puzzle:

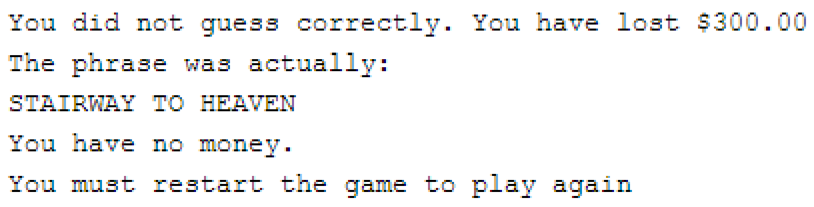
Input the phrase you think is the answer. You do not have to capitalize, but you do need to correctly match all the letters and spaces. If you incorrectly guess, you will lose $300.00; if you correctly guess, you will gain 30 points for each hidden letter revealed. You will then be displayed the amount of money left in your account and current amount of points you have earned.



```
Song Title
ST□□R□□□ T□ H□□V□□
Your keyboard:
A■CDEFG■IJ■LM
NOPQ■■■■■WXYZ
Input the final answer: stairway to heaven
You gain 30 points for each hidden letter you guessed
You gain 300 points
Congrats you win!
You have $50.00 left in your account
Your score: 600 points
```

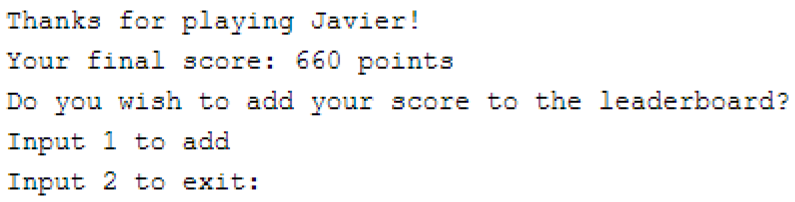
Losing and Leaderboard:

If you run out of money, the correct phrase will be displayed, you then lose the game and have to exit the program. But you will have an option to enter your score into the leaderboard.



```
You did not guess correctly. You have lost $300.00
The phrase was actually:
STAIRWAY TO HEAVEN
You have no money.
You must restart the game to play again
```

If you won, you can exit the program through the menu and still have a chance to enter your score to the leaderboard as well.



```
Thanks for playing Javier!
Your final score: 660 points
Do you wish to add your score to the leaderboard?
Input 1 to add
Input 2 to exit:
```

Have fun playing!

Project Summary

Project size	**557 Lines**
Lines of code	473 Lines
Comment lines	63 Lines
Blank lines	21 Lines

This project was slightly more difficult than I expected, but it was very manageable. Everything featured in the program was intended from preproduction planning. I originally planned to include a lot of asci graphics from files, but it proved to be too time consuming. The other feature I decided to cut was player continuation. I deemed it unimportant and not that necessary for the game. You may see the development process in my GitHub repository. Included are all previous versions and a bonus binary project I created for testing purposes. Running code from previous versions may not work and is not advised.

https://github.com/javierborja95/JB_CSC17a/tree/master/Project

Version 1

Main is developed along with a function that displays the menu. Functions are created to develop the library and read its contents. A header file is created and includes structures of future variables.

Version 2

The bulk of the program is developed in this stage. A game function is mostly completed with its necessary functions required for play.

Version 3

The point and money system is finalized in this version. The ability to write and read scores to binary files are possible and a leaderboard function to accompany these new features are developed. Comments are added to increase readability.

Version 4

Testing to catch possible glitches in the game are completed. There are very minor changes from version 3, but the game is finalized in this stage.

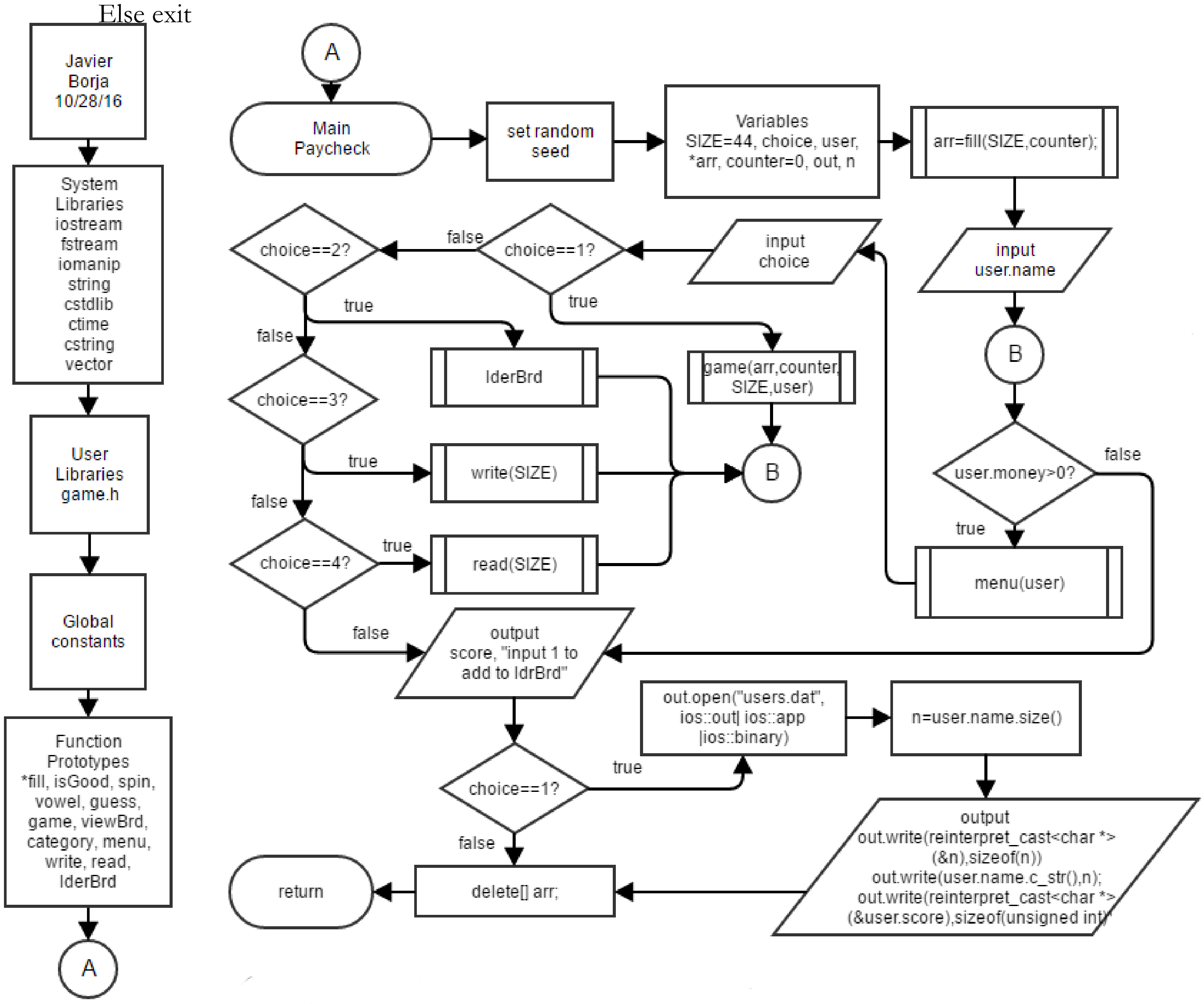
Pseudocode and Flowcharts

Function prototypes:

```
unsigned int* fill(int,int&)
bool isGood(char[],int)
bool spin(Letter*,Letter*,int,int,Player&,Clue)
bool vowel(Letter*,Letter*,int,int,Player&,Clue)
bool guess(Letter*,Letter*,int,int,Player&,Clue)
void game(unsigned int*,int,int,Player&)
void viewBrd(Letter*,Letter*,int,int,Player)
void categry(int)
void menu(Player)
void write(int)
void read(int)
void lderBrd()
```

Main:

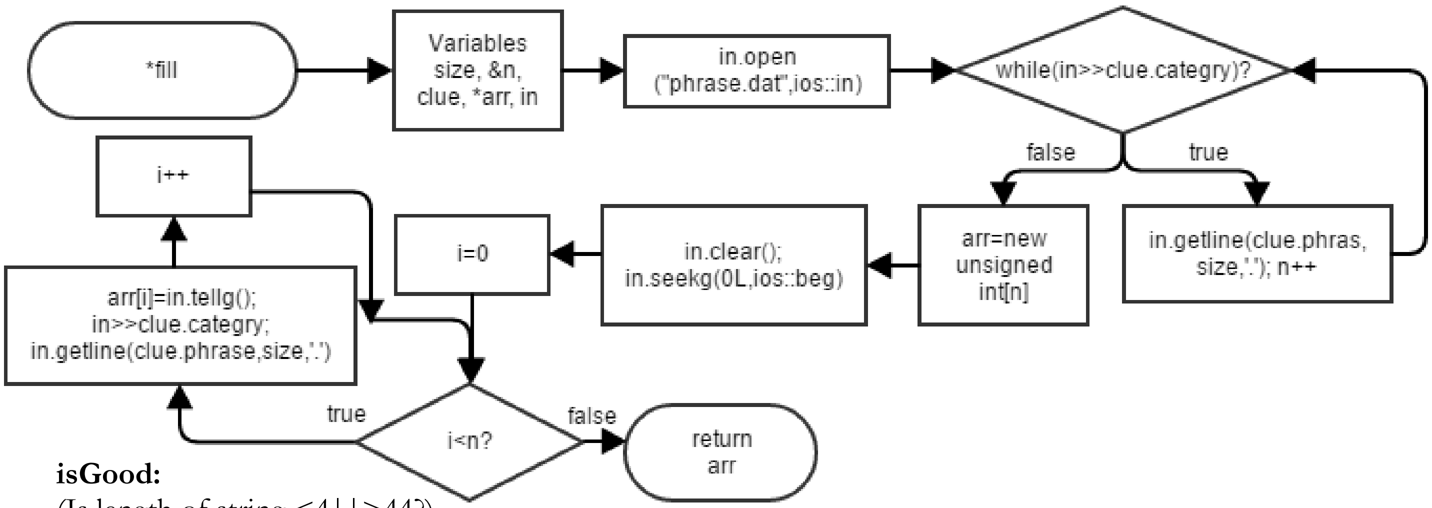
```
Set Random Number seed
Get array of phrases
Input name
Do{
    Output money and score
    menu()
    Input choice
    Switch(choice){
        Case 1: game()
        Case 2: lderBrd()
        Case 3: write()
        Case 4: read()
        }
}While(choice 1-4 && money>0)
Ask to write score to leaderboard
    Write scores to file
    Else exit
```



Javier Borja 10/28/16
System Libraries iostream fstream iomanip string cstdlib ctime cstring vector
User Libraries game.h
Global constants
Function Prototypes *fill, isGood, spin, vowel, guess, game, viewBrd, category, menu, write, read, lderBrd
A
A
Main Paycheck
set random seed
Variables SIZE=44, choice, user, *arr, counter=0, out, n
arr=fill(SIZE,counter);
input user.name
B
user.money>0?
false
true
menu(user)
input choice
choice==1?
false
true
game(arr,counter, SIZE,user)
B
choice==2?
true
false
lderBrd
choice==3?
true
false
write(SIZE)
choice==4?
true
false
read(SIZE)
output score, "input 1 to add to ldrBrd"
choice==1?
true
false
out.open("users.dat", ios::out| ios::app |ios::binary)
n=user.name.size()
output out.write(reinterpret_cast<char *>(&n),sizeof(n)) out.write(user.name.c_str(),n); out.write(reinterpret_cast<char *>(&user.score),sizeof(unsigned int))
delete[] arr;
return

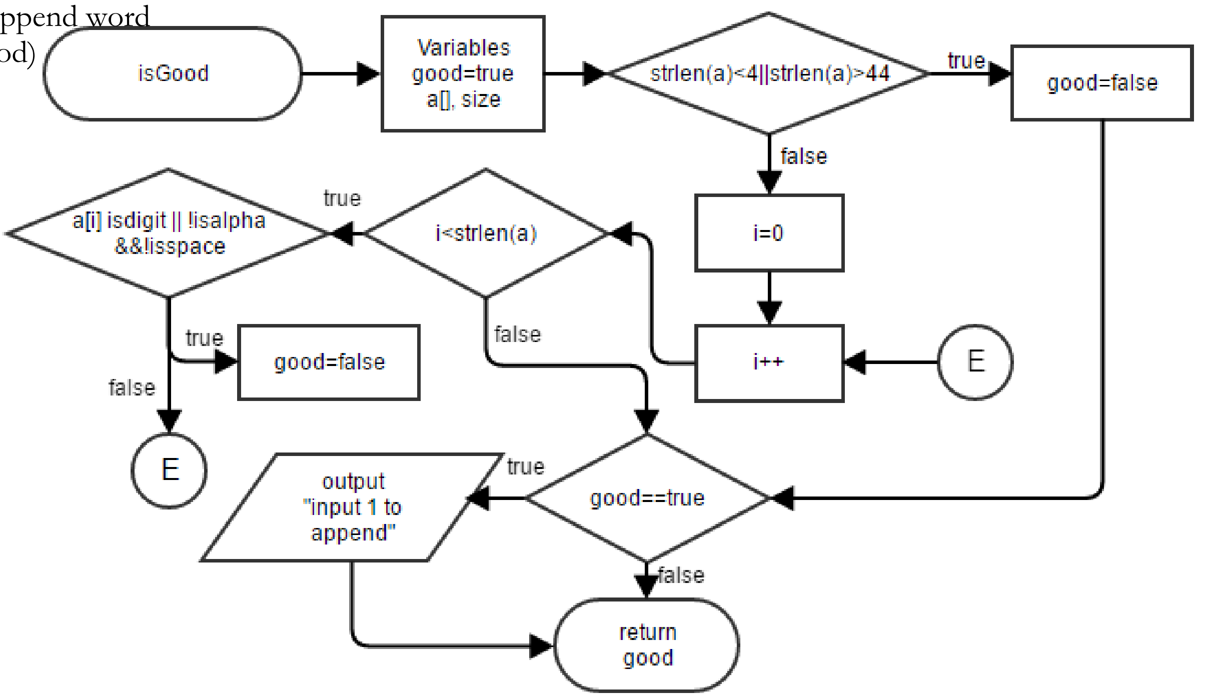
Fill:

```
Open phrase file
While(get category)
    Get clue phrase
    Size of array++
Allocate memory by size of array
Seek to beginning of file
For(i=0; i<size of array; i++){
    While(get category)
        Get clue phrase
        Set position of index
} return array
```


*fill
Variables size, &n, clue, *arr, in
in.open ("phrase.dat",ios::in)
while(in>>clue.categry)?
false
true
in.getline(clue.phras, size,'.'); n++
arr=new unsigned int[n]
in.clear(); in.seekg(0L,ios::beg)
i=0
i++
arr[i]=in.tellg(); in>>clue.categry; in.getline(clue.phrase,size,'.')
true
i<n?
false
return arr

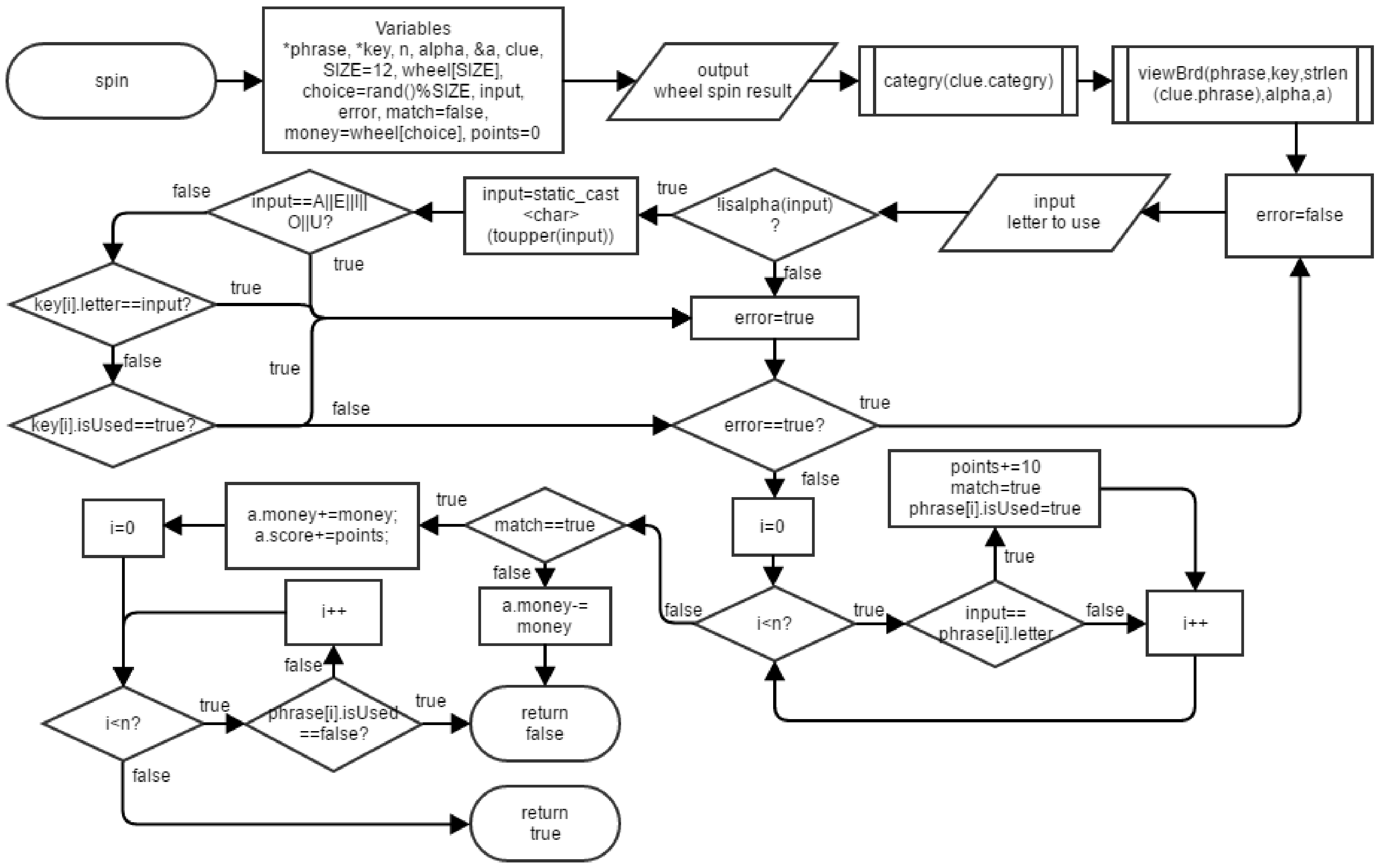
isGood:

```
(Is length of string <4||>44?)
T: Good=false
F: For(i=0;i<length of string;i++){
    (If character is not letter or space) good=false
If(good==true)
T: Ask to append word
Return (good)
```


isGood
Variables good=true a[], size
strlen(a)<4||strlen(a)>44
true
good=false
false
i=0
i<strlen(a)
true
a[i] isdigit || !isalpha &&!isspace
true
good=false
false
E
false
i++
E
output "input 1 to append"
true
good==true
false
return good

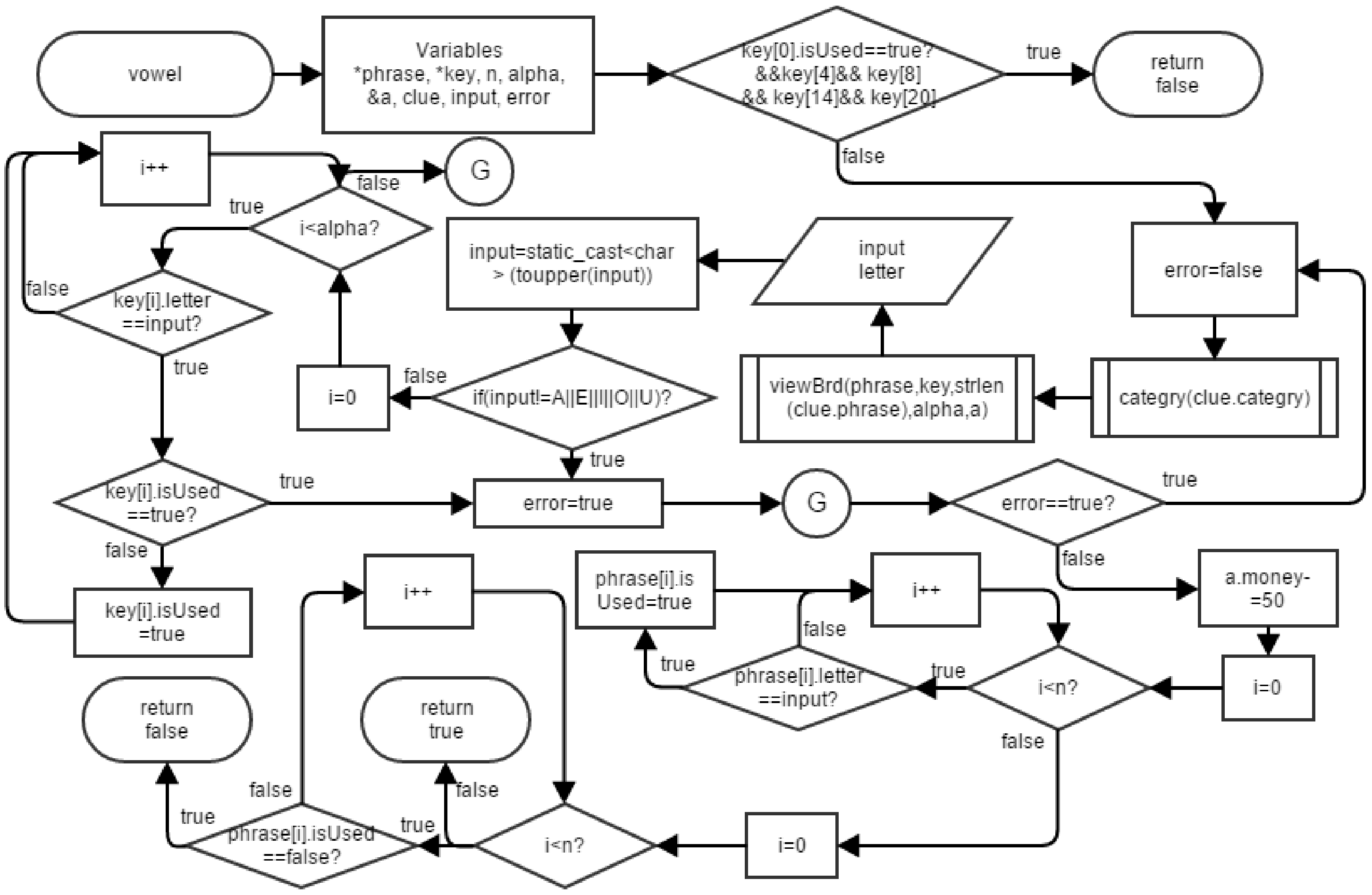
Spin:

```
Spin wheel
Show board and keyboard
Do{
    Error=false
    Input letter to use
    (is letter==vowel||non alphabet||or already used?)
        Error=true
While(error==true)
If(letter input==hidden letter)
    T: {add points
    Match=true}
If(match==true)
    T:{Add money
    Add points to score
    Make hidden letters shown
    (If all letters revealed) Return win
    }
    F: lose money, return loss
```

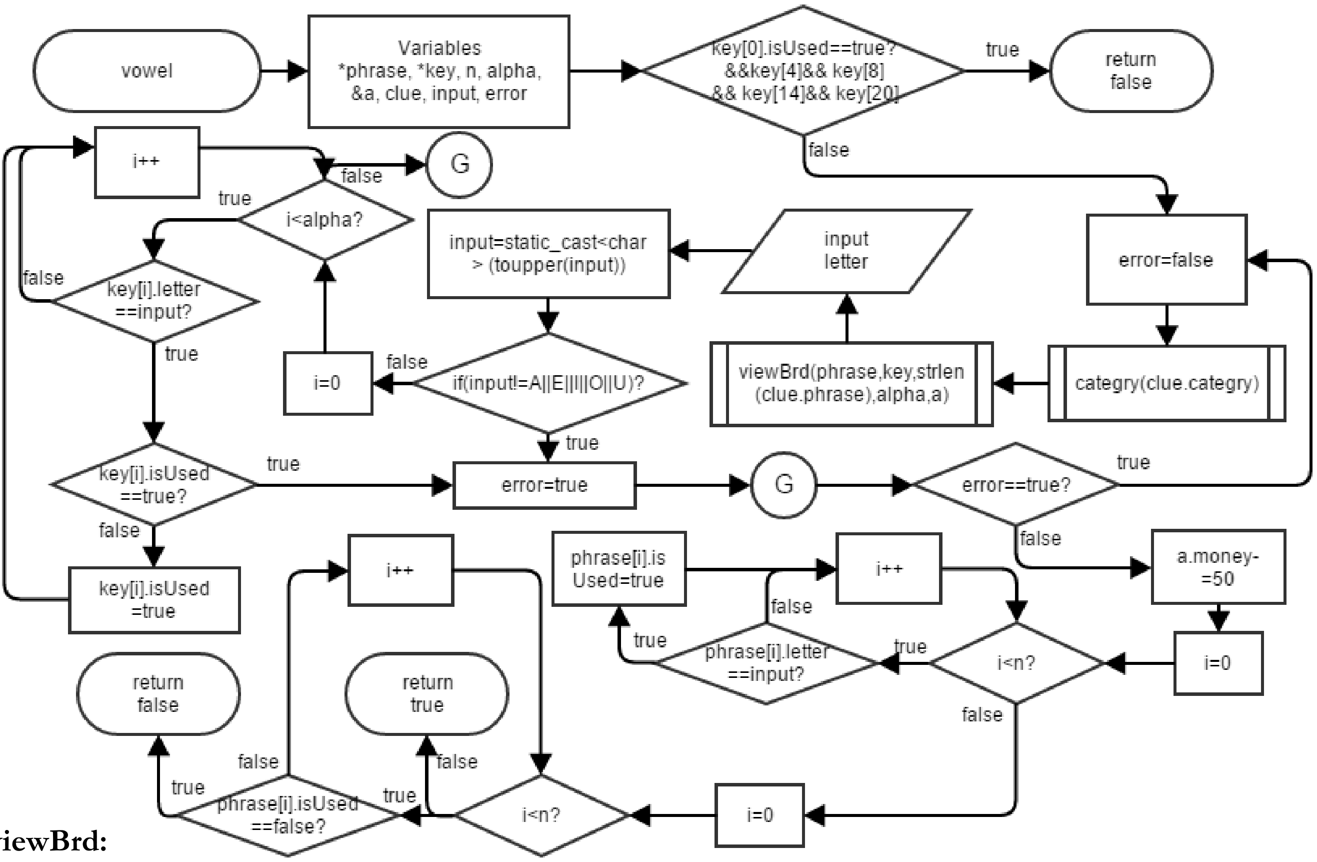


Vowel:

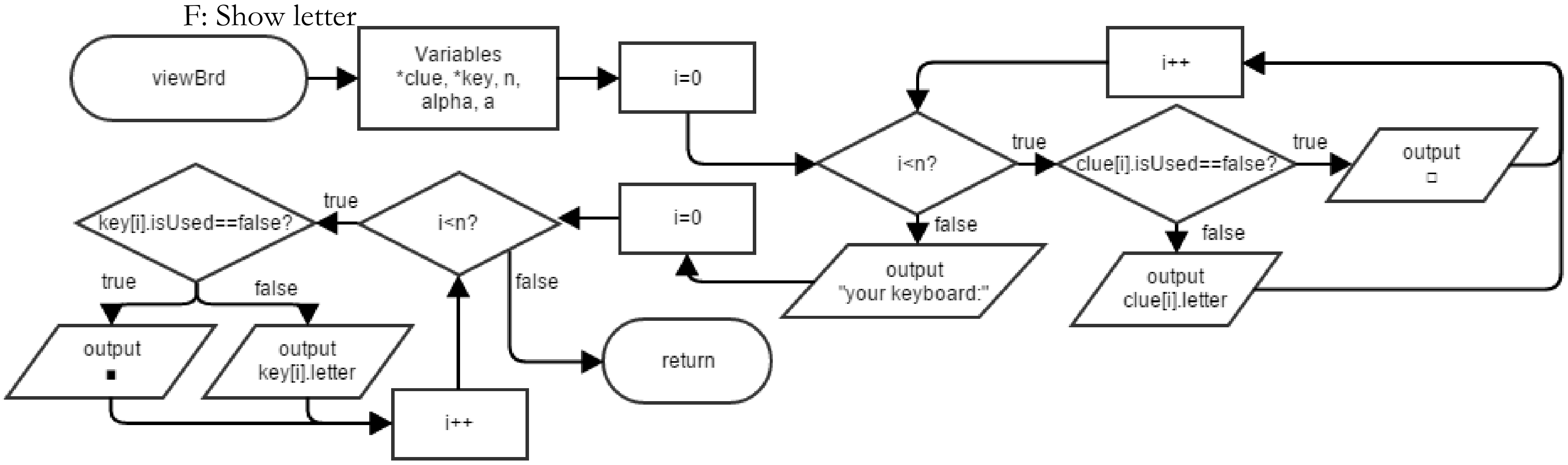
```
(if all vowels are used) return
Do{
   Error =false
   Show board and keyboard
   Input letter
   (if input is not vowel)
      Error =true
   (if vowel is used)
      T: error =false
      F: make key used
While(error==true)
Subtract money
Reveal vowels from phrase
(If all letters are revealed) return win
Else return loss
```



Guess:
Show board and keyboard
Input phrase
(if input matches board phrase)
    T: Return win
    F: Subtract money, return loss

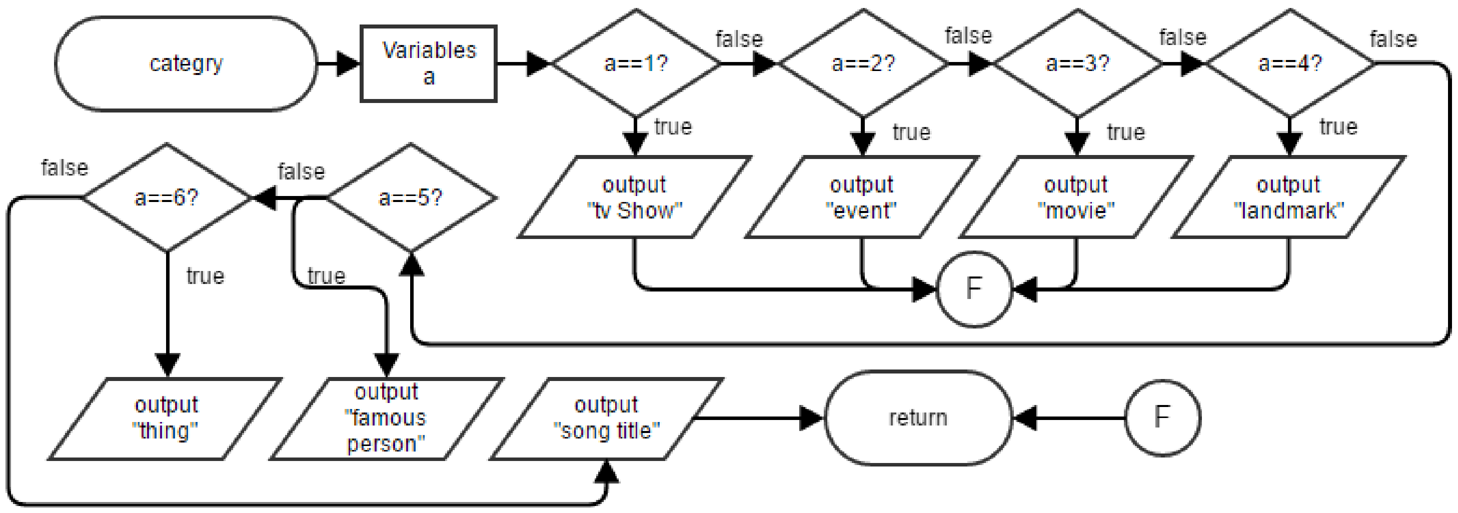

vowel
Variables *phrase, *key, n, alpha, &a, clue, input, error
key[0].isUsed==true? &&key[4]&& key[8] && key[14]&& key[20]
true
return false
false
error=false
categry(clue.categry)
viewBrd(phrase,key,strlen (clue.phrase),alpha,a)
input letter
input=static_cast<char > (toupper(input))
if(input!=A||E||I||O||U)?
false
true
i=0
i<alpha?
i++
G
key[i].letter ==input?
key[i].isUsed ==true?
key[i].isUsed =true
error=true
error==true?
a.money- =50
i<n?
phrase[i].letter ==input?
phrase[i].is Used=true
i++
phrase[i].isUsed ==false?
return false
return true

viewBrd:
(If phrase letter is hidden)
    T: Show square
    F: Show letter
(If keyboard letter is used)
    T: Show square
    F: Show letter



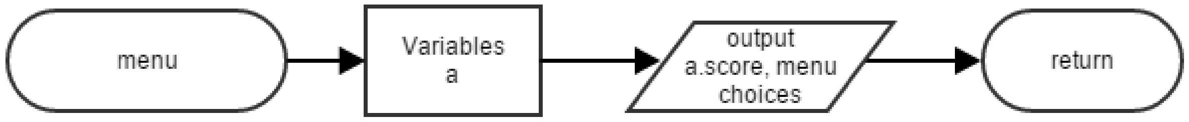
Categry:

Switch(number)
  Display category based on number



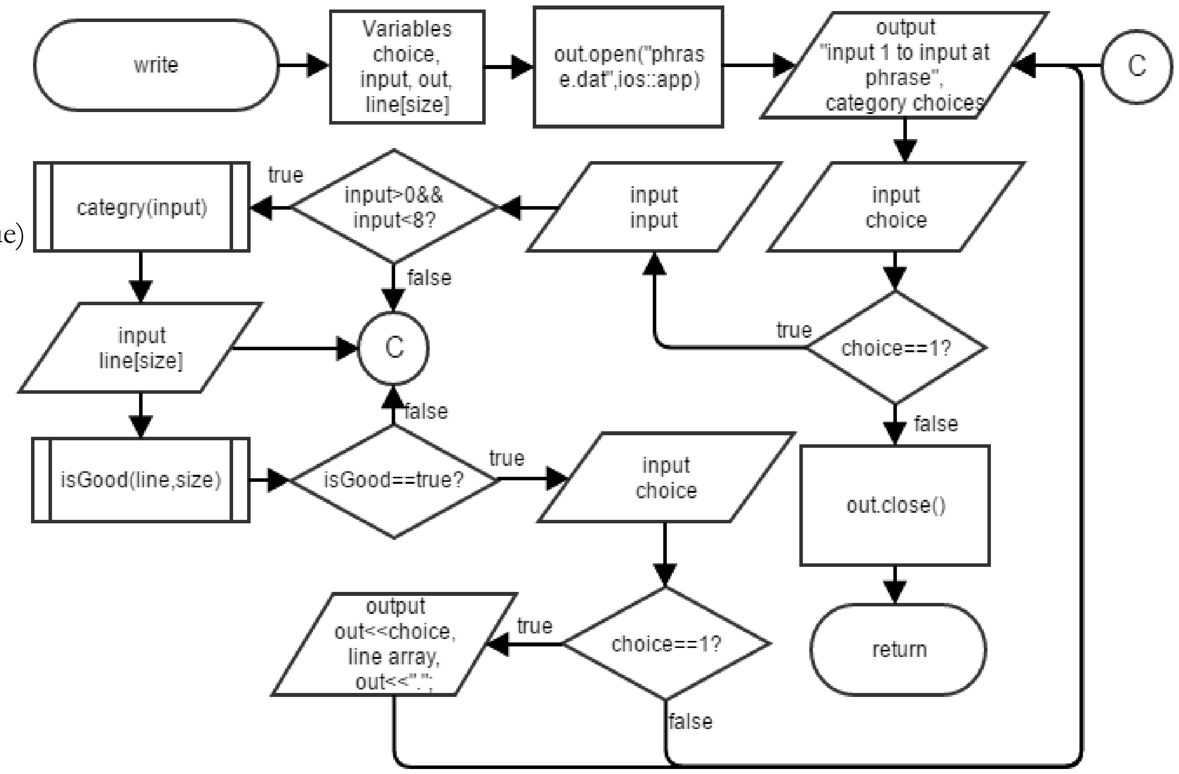
Menu:

Display menu contents



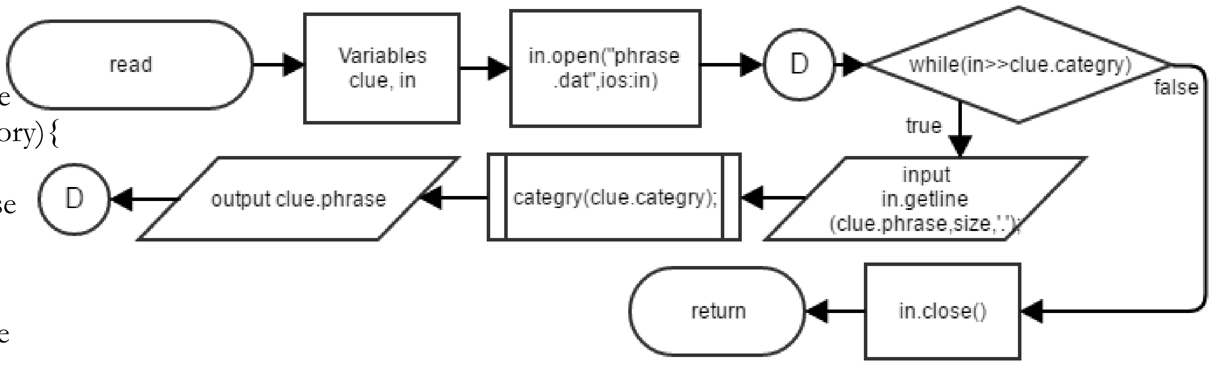
Write:

```
Open file to append
Input category
Input phrase
isGood(phrase)
(if good)
    T:Ask to input
    Input choice
    (If choice is true)
        write to file
Close file
```


write
Variables choice, input, out, line[size]
out.open("phrase.dat",ios::app)
output "input 1 to input at phrase", category choices
C
input choice
choice==1?
true
false
input input
input>0&& input<8?
categry(input)
input line[size]
isGood(line,size)
isGood==true?
input choice
choice==1?
output out<<choice, line array, out<<".";
out.close()
return

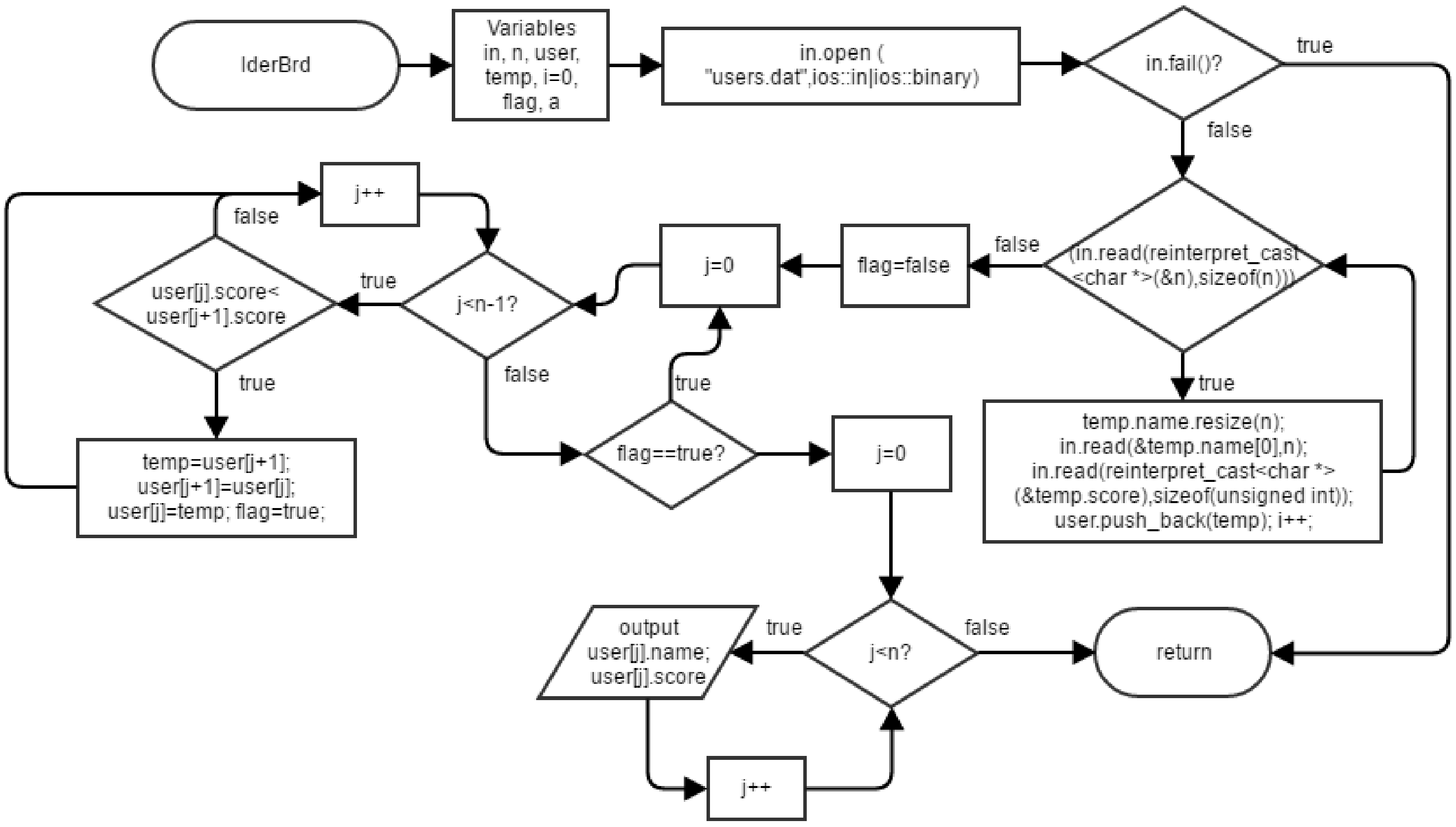
Read:

```
Open phrase file
(while get category){
    Categry()
    Output phrase
} close file
```



lderBrd:

```
Open Binary file
If(fail) return
While(reading size of string){
    Get name string and score
    Push back array with name and score
}do{
    For(i=0;i<number of elements in array; i++){
        Flag=false
        If (element i score is greater>than element i+1 score)
            Swap
            Flag=true
}while(flag=true)
Display sorted names and scores
```


lderBrd
Variables in, n, user, temp, i=0, flag, a
in.open ("users.dat",ios::in|ios::binary)
in.fail()?
true
false
(in.read(reinterpret_cast <char *>(&n),sizeof(n)))
false
true
temp.name.resize(n); in.read(&temp.name[0],n); in.read(reinterpret_cast<char *> (&temp.score),sizeof(unsigned int)); user.push_back(temp); i++;
flag=false
j=0
j<n-1?
true
false
user[j].score< user[j+1].score
false
true
temp=user[j+1]; user[j+1]=user[j]; user[j]=temp; flag=true;
j++
flag==true?
true
j=0
j<n?
true
false
output user[j].name; user[j].score
j++
return

Major Variables:

Type	Variable Name	Description	Location
struct	Player	Contains user name, money and score	main, spin, vowel, guess, viewBrd, menu, lderBrd
	Clue	Contains category and phrase	spin, vowel, guess
	Letter	Contains a letter and bool of use status	spin, vowel, guess, game, viewBrd
vector<Player>	user	Array of Player structures	lderBrd
bool	good	Flag to check if input is good	isGood
	isUsed	If letter is used	Letter
	win	If win, win=true, else win=false	game
	error	Flag to check if input is correct	spin, vowel
	match	Flag to see if a letter matches	spin
	flag	Flag to stop loop	lderBrd
unsigned int	*arr	Index array to find categories and clues	main, fill, game
	srand	Random number	main, game
	score	Player score	Player
	categry	Clue category	Clue
int	SIZE	Size constant	main,
	counter	Number of phrases in dictionary	main, fill, game
	wheel[SIZE]	Array of options a player can spin	spin
	money	Money to add or subtract from player	game, spin, guess, vowel
	points	Points to add to player	game, spin, guess, vowel
char	phrase[44]	Clue phrase	Clue
	letter	Letter in Letter struct	Letter
	choice	Menu choice	main, write
	input	Sub menu choice	write, spin, vowel
	line[size]	Character array to add to dictionary	write, isGood
	option	Option for game menu	game
fstream	in	Used for input	read, fill, game, lderBrd
	out	Used for output	main, write
string	name	Player name	Player
Player	user	User	main, spin, vowel, guess, game,viewBrd,categry,menu
Clue	clue	Clue to display	read, fill, game, viewBrd, spin, vowel, guess
Letter	*phrase	Struct array of Letters for clue phrase	game, spin, vowel, guess
	*kyBoard	Struct array of Letters for alphabet array	game, spin, vowel, guess

Concepts utilized:

Project requirements listed by chapter, with bonus vector construct.

Tony Gaddis, Starting out with C++ *From Control Structures through Objects*, Eighth Edition.

Chapter	Construct	Location
9.2	Pointer Variables	unsigned int *arr;
9.3	Pointer Arrays	arr=new unsigned int[n];
9.6	Comparing Pointers	if(user[j].score<user[j+1].score)
9.7	Pointers as Function Parameters	void viewBrd(Letter*,Letter*,int,int,Player);
9.8	Dynamic Memory Allocation	phrase=new Letter[strlen(clue.phrase)];
9.9	Returning Pointers from Functions	unsigned int* fill(int size,int &n){… return arr;
10.1	Character Testing	if(isdigit(a[i])||(!isalpha(a[i])&&!isspace(a[i])))
10.2	Character Conversion	input=static_cast<char>(toupper(input));
10.3	C-string arrays	char line[size];
10.4	Library Functions for Working with C-Strings	for(int i=0;i<strlen(clue.phrase);i++){
10.6	Writing your Own C-String-Handling Functions	bool isGood(char a[],int size){
10.7	More About the C++ String Class	in.getline(clue.phrase,size,'.');
11.1	Abstract Data Types	struct Letter{ char letter; bool isUsed=false; };
11.2	Combining Data into Structures	struct Player{ std::string name; int money=50; unsigned int score=0; };
11.3	Accessing Structure Members	cout<<clue[i].letter;
11.4	Initializing a Structure	struct Clue{ unsigned int category; phrase[44]; };
11.5	Arrays of Structures	Letter *phrase,*kyBoard;
11.7	Structures as Function Arguments	bool spin(Letter*,Letter*,int,int,Player&,Clue);
11.8	Pointers to Structures	Letter *phrase,*kyBoard;
12.1	File Operations	fstream in;
12.2	File Output Formatting	cout<<setw(5)<<right<<user[j].score<<" points"<<endl<<endl;
12.4	Error Testing	if(in.fail()){
12.5	Member Functions for Reading and Writing Files	in.getline(clue.phrase,size,'.');
12.7	Binary Files	in.read(reinterpret_cast<char*>(&temp.score), sizeof(unsigned int));
12.8	Creating Records with Structures	out.write(reinterpret_cast<char *>(&n),sizeof(n)); out.write(user.name.c_str(),n); out.write(reinterpret_cast<char *>(&user.score), sizeof(unsigned int));
12.9	Random-Access Files	in.seekg(a[index],ios::beg);
8.5	Searching and Sorting Vectors	vector<Player> user; temp.name.resize(n); in.read(&temp.name[0],n); in.read(reinterpret_cast <char *> (&temp.score),sizeof(unsigned int)); user.push_back(temp); …if(user[j].score<user[j+1].score)…

References:

Gaddis, Tony. Starting out with C++ from Control Structures through Objects. 8th ed. Pearson Addison-Wesley, 2014. Print.

http://wheeloffortuneanswer.com/
Copied phrases to fill dictionary.

Code:

```
#ifndef GAME_H
#define GAME_H

struct Player{
   std::string name;
   int money=50;       //Player starts with $500.00
   unsigned int score=0;//Player starts with 0 points
};

struct Clue{
   unsigned int categry;
   char phrase[44];    //Max Phrase length
};

struct Letter{
   char letter;
   bool isUsed=false;
};

#endif /* GAME_H */

//System Libraries
#include <iostream>  //Input/ Output Stream Library
#include <fstream>   //File I/O
#include <iomanip>   //Output Manipulation
#include <string>    //Strings
#include <cstdlib>   //Random Library
#include <ctime>     //Time Library
#include <cstring>   //Cstrings for strlen() function
#include <vector>    //Vectors
using namespace std; //Namespace of the System Libraries

//User Libraries
#include "game.h"
//Global Constants

//Function Prototypes
unsigned int* fill(int,int&);
bool isGood(char[],int);
bool spin(Letter*,Letter*,int,int,Player&,Clue);
bool vowel(Letter*,Letter*,int,int,Player&,Clue);
bool guess(Letter*,Letter*,int,int,Player&,Clue);
void game(unsigned int*,int,int,Player&);
void viewBrd(Letter*,Letter*,int,int,Player);
void category(int);
void menu(Player);
void write(int);
void read(int);
void lderBrd();
//Execution

int main(int argc, char** argv) {
    //Set Random seed
    srand(static_cast<unsigned int>(time(0)));
    //Variables
    const int SIZE=44; //Max size of char array
    char choice;       //Menu Choice
    Player user;       //User
    unsigned int *arr; //Index array to find categories and clues
    int counter=0;     //Index starts at zero
    fstream out;       //Output stats to file
    int n;             //Size of string if output to file

    //Input Data
    arr=fill(SIZE,counter);
    cout<<"Input your name: ";
    getline(cin,user.name);
    cout<<"Welcome to Wheel of Fortune "<<user.name<<"!\n"
            "Your money = $"<<user.money<<"0.00\n";
    do{
        menu(user);
        cin>>choice;
        cin.ignore();
    //Process Data
        switch(choice){
            case'1':
                game(arr,counter,SIZE,user);
                break;
            case'2':
                lderBrd();
                cout<<"Your money = $"<<user.money<<"0.00\n";
                break;
            case'3':
                write(SIZE);
                break;
            case'4':
                read(SIZE);
                break;
        }
    }while((choice=='1'||choice=='2'||choice=='3'||choice=='4')&&user.money>0);
    //Output Data
    cout<<"Thanks for playing "<<user.name<<"!"<<endl;
    cout<<"Your final score: "<<user.score<<" points"<<endl;
    cout<<"Do you wish to add your score to the leaderboard?\n"
            "Input 1 to add\n"
            "Input 2 to exit: ";
    cin>>choice;
    if(choice=='1'){
        out.open("users.dat",ios::out|ios::app|ios::binary);
        n=user.name.size();
        out.write(reinterpret_cast<char *>(&n),sizeof(n));
        out.write(user.name.c_str(),n);
        out.write(reinterpret_cast<char *>(&user.score),sizeof(unsigned int));
        cout<<"Your score has been added"<<endl;
    }
    //Deallocate Memory
    delete[] arr;

    return 0;
}

void menu(Player a){
    //Output Data
            cout<<"Your score: "<<a.score<<" points\n\n"
            "Select an option below:\n"
            " 1. Begin a new game of Wheel of Fortune\n"
            " 2. View the leaderboard\n"
            " 3. Append to the Library\n"
            " 4. View the Library(You'll spoil all the answers!)\n\n"
            "Any other input to exit: ";
}

void write(int size){
    //Variables
    char choice;    //Menu choice
    char input;     //Input for sub-menu
    fstream out;    //Output to file
    char line[size];//Character array of size=44

    //Open File
    out.open("phrase.dat",ios::app);
    //Input Data
    do{
        cout<<endl<<"Input 1 to input a phrase\n"
                "Input 0 to exit: ";
        cin>>choice;
        cin.ignore();
        if(choice=='1'){
            cout<<"Input a category:\n"
                    "1 TV Show \n"
                    "2 Event \n"
                    "3 Movie \n"
                    "4 Landmark \n"
                    "5 Famous Person \n"
                    "6 Thing \n"
                    "7 Song Title \n\n"
                    "0 Exit: ";
            cin>>input;
            cin.ignore();
    //Output Data
            if(input>48&&input<56){ //If input is '1'-'7'
                categry(input-48);
                cout<<"Input your phrase(max 44 characters): "<<endl;
                cin.getline(line,size);
                if(isGood(line,size)){ //If input is good ask if wish to append
                    cin>>choice;
                    cin.ignore();
                    if(choice=='1'){
                        out<<input;
                        for(int i=0;i<strlen(line);i++){
                            out<<static_cast<char>(toupper(line[i])); //Make uppercase
                        }
                        out<<"."<<endl;
                        cout<<"You must restart the game for effects to take effect"<<endl;
                    }
                }
            }
        }
    }while(choice=='1');
    //Close File
    out.close();
}

void read(int size){
    //Variables
    Clue clue; //Clue
    fstream in;//Input from file
    //Open File
    in.open("phrase.dat",ios::in);
    //Input Data
    while(in>>clue.categry){ //While in can still get information
        in.getline(clue.phrase,size,'.');
    //Output Data
        categry(clue.categry);
        cout<<clue.phrase<<endl;
    }
    //Close Files
    in.close();
    cout<<endl;
}

bool isGood(char a[],int size){
    //Variables
    bool good=true;
    //Process Data
    if(strlen(a)<4||strlen(a)>44){ //If char array doesn't fit size limit
        good=false;              //Return not good
        cout<<"ERROR: Phrase must be greater than 3 characters and less than 44"<<endl;
    }
    for(int i=0;i<strlen(a);i++){
        if(isdigit(a[i])||(!isalpha(a[i])&&!isspace(a[i]))){//If not space or letter
            cout<<"ERROR: Input must be characters only\n";
            good=false;
        }
        if(!good)break;
    }
    //Output Data
    if(good){
        cout<<"Do you really wish to add the following phrase?"<<endl;
        for(int i=0;i<strlen(a);i++){
            cout<<static_cast<char>(toupper(a[i]));
        }
```

```
        cout<<endl<<endl<<"Input 1 to append\n"
                "Or anything else to cancel: ";
    }
    return good;
}

unsigned int* fill(int size,int &n){
    //Variables
    Clue clue;        //Temp clue to put info in
    unsigned int *arr;//Array of positions in phrase file
    fstream in;       //Input
    //Open File
    in.open("phrase.dat",ios::in);
    //Input Data
    while(in>>clue.categry){
        in.getline(clue.phrase,size,'.');
        n++;          //Add to size of array
    }
    //Allocate Memory
    arr=new unsigned int[n];
    in.clear();
    in.seekg(0L,ios::beg); //Go back to beginning of file
    //Process Data
    for(int i=0;i<n;i++){
        arr[i]=in.tellg(); //Each index has a position
        in>>clue.categry;
        in.getline(clue.phrase,size,'.');
    }
    //Output Data
    return arr;
}

void game(unsigned int *a,int i,int size,Player &user){
    //Variables
    fstream in;           //Input
    Clue clue;            //Clue with category and phrase
    int index=(rand()%i); //Index to choose clue
    bool win=false;       //Win bool
    Letter *phrase,*kyBoard; //Pointer arrays of clue phrase and alphabet
    const int SIZE=26;    //Size of kyBoard array
    char option;          //Option to spin,buy,or guess
    //Open File
    in.open("phrase.dat",ios::in);
    //Input Data
    in.seekg(a[index],ios::beg); //Go to position in file to get phrase and clue
    in>>clue.categry;
    in.getline(clue.phrase,size,'.');
    //Process Data
    //Allocate Memory
    kyBoard=new Letter[SIZE];             //New Array of Letter for keyboard
    phrase=new Letter[strlen(clue.phrase)];//New Array of Letter for phrase
    for(int j=0;j<SIZE;j++){     //Initialize the alphabet
        kyBoard[j].letter='A'+j;
    }
    for(int j=0;j<strlen(clue.phrase);j++){//Initialize phrase array with clue
        phrase[j].letter=clue.phrase[j];
        if(isspace(phrase[j].letter)){     //If letter is space
            phrase[j].isUsed=true;         //Don't hide it
        }
    }
    //Process Data
    do{
        categry(clue.categry);
        viewBrd(phrase,kyBoard,strlen(clue.phrase),SIZE,user);
        do{
            cout<<"Your money = $"<<user.money*10<<".00"<<endl;
            cout<<endl<<endl<<"What would you like to do?"<<endl;
            cout<<  " 1. Spin the Wheel ✪\n"
            " 2. Buy a vowel ($500.00)\n"
            " 3. Solve the Puzzle 🖂(Bad guess lose $300.00)\n"<<endl;
            cin>>option;
            cin.ignore();
            switch(option){
                case'1':
                    win=spin(phrase,kyBoard,strlen(clue.phrase),SIZE,user,clue);
                    break;
                case'2':
                    if(user.money<=50){
                        cout<<"You don't have enough money!"<<endl;
                        cout<<"Spin the wheel or guess the puzzle"<<endl;
                        break;
                    }
                    win=vowel(phrase,kyBoard,strlen(clue.phrase),SIZE,user,clue);
                    break;
                case'3':
                    win=guess(phrase,kyBoard,strlen(clue.phrase),SIZE,user,clue);
                    if(win==false){
                        cout<<"You did not guess correctly. You have lost $300.00\n";
                        user.money-=30;
                    }
                    break;
                default: cout<<"ERROR: Bad Input"<<endl;
            }
        }while(option<49||option>51);
    }while(win==false&&user.money>0); //Loop until win or lose
    //Output Data
    if(user.money<=0){
        cout<<"The phrase was actually: "<<endl;
        for(int j=0;j<strlen(clue.phrase);j++){
            cout<<clue.phrase[j];
        }cout<<endl;
        cout<<"You have no money.\n"
            "You must restart the game to play again"<<endl;
    }else cout<<"Congrats you win!\n"
            "You have $"<<user.money*10<<".00 left in your account"<<endl;
    //Deallocate Memory
    delete[] phrase;
    delete[] kyBoard;
    //Close File
    in.close();
}
```

```
void category(int a){
    //Output Data
    switch(a){
            case 1:
                cout<<"TV Show"<<endl;
                break;
            case 2:
                cout<<"Event"<<endl;
                break;
            case 3:
                cout<<"Movie"<<endl;
                break;
            case 4:
                cout<<"Landmark"<<endl;
                break;
            case 5:
                cout<<"Famous Person"<<endl;
                break;
            case 6:
                cout<<"Thing"<<endl;
                break;
            default:
                cout<<"Song Title"<<endl;
    }
}

void viewBrd(Letter *clue,Letter *key,int n,int alpha,Player a){
    //Output Data
    for(int i=0;i<n;i++){         //Go through clue array
        if(clue[i].isUsed==false){//If letter has not been used, hide letter
            cout<<"□";
        }else{
            cout<<clue[i].letter;
        }
    }
    cout<<endl<<"Your keyboard:"<<endl;
    for(int i=0;i<alpha;i++){    //Go through keyboard array
        if(key[i].isUsed==false){//If letter has not been used, hide letter
            cout<<key[i].letter;
        }else cout<<"■";
        if((i+1)%13==0) cout<<endl;
    }
}

bool spin(Letter *phrase,Letter *key,int n,int alpha,Player &a,Clue clue){
    //Variables
    const int SIZE=12;     //Possible Options of wheel
    int wheel[SIZE]={0,0,5,5,10,15,15,20,25,30,35,40};
    int choice=rand()%SIZE;//Random wheel choice
    char input;            //Letter input
    bool error;            //Incorrect letter input
    bool match=false;      //Did letter match?
    int money=wheel[choice];//Money to add or subtract from user's money
    int points=0;          //Counter for points
    //Input Data
    cout<<"Spinning...\nPress Enter to continue";
```

```
cin.get();
cout<<"__________________________________________"<<endl;
if(money==0) cout<<"You spun a free guess"<<endl;
else cout<<endl<<"You spun $"<<money*10<<".00"<<endl;
categry(clue.categry);
viewBrd(phrase,key,strlen(clue.phrase),alpha,a);
do{
   error=false;
   cout<<"What letter do you want to use? ";
   cin>>input;
   cin.ignore();
//Process Data
   if(!isalpha(input)){
      cout<<"Input must be part of the alphabet"<<endl;
      error=true;
   }
   input=static_cast<char>(toupper(input)); //Make uppercase
   if(input=='A'||input=='E'||input=='I'||input=='O'||input=='U'){
      cout<<"You have to buy vowels"<<endl;
      error=true;
   }
   if(!error){
      for(int i=0;i<alpha;i++){
         if(key[i].letter==input){
            if(key[i].isUsed==true){
               cout<<"You already used that letter"<<endl;
               return false;
            }else key[i].isUsed=true;
         }
      }
   }
}while(error); //Keep looping until valid input
for(int i=0;i<n;i++){
   if(input==phrase[i].letter){//If letter matches
      points+=10;          //Add ten points
      match=true;          //Match is true
      phrase[i].isUsed=true;  //Don't hide letter anymore
   }
}
//Output Data
if(match){ //If match is true
   cout<<"You have been awarded $"<<money*10<<".00"<<endl;
   a.money+=money;
   cout<<"You gain 10 points for each letter guessed"<<endl;
   cout<<"You gained "<<points<<" points"<<endl;
   a.score+=points;
   for(int i=0;i<n;i++){
      if(phrase[i].isUsed==false){
         return false;//Not all letters are revealed, win=false;
      }
   }
   return true;       //All letters of phrase are revealed, win=true
}else{ //Match is not true
   a.money-=money;
   cout<<"__________________________________________"<<endl;
   cout<<"You have lost $"<<money*10<<".00."<<endl;
```

```
    }

    return false;

}

bool vowel(Letter *phrase,Letter *key,int n,int alpha,Player &a,Clue clue){
    //Variables
    char input; //Input for vowel
    bool error; //Error
    //Input Data
    if((key[0].isUsed)&&(key[4].isUsed)&&(key[8].isUsed)&&(key[14].isUsed)&&(key[20].isUsed)){
        cout<<"You have already bought all the vowels"<<endl;
        return false;
    }
    do{
        error=false;
        cout<<"__________________________________________"<<endl;
        categry(clue.categry);
        viewBrd(phrase,key,strlen(clue.phrase),alpha,a);
        cout<<"Which vowel do you want to buy? ";
        cin>>input;
        cin.ignore();
        input=static_cast<char>(toupper(input));
        if(input=='A'||input=='E'||input=='I'||input=='O'||input=='U'){

        }
        else{
            cout<<"You did not choose a vowel"<<endl;
            error=true;
        }
        if(error==false){
            for(int i=0;i<alpha;i++){
                if(key[i].letter==input){
                    if(key[i].isUsed==true){
                        cout<<"You already used that letter"<<endl;
                        error=true;
                    }else key[i].isUsed=true;
                }
            }
        }
    }while(error==true); //Loop until valid input

    //Process Data
    cout<<"You have bought a vowel for $500.00"<<endl;
    a.money-=50; //Subtract money from user
    for(int i=0;i<n;i++){
        if(phrase[i].letter==input) //Reveal vowels from clue phrase
            phrase[i].isUsed=true;
    }
    for(int i=0;i<n;i++){
        if(phrase[i].isUsed==false){
            return false;//Not all letters are revealed, win=false;
        }
    }
    return true;         //All letters of phrase are revealed, win=true
```

```
}

bool guess(Letter *phrase,Letter *key,int n,int alpha,Player &a,Clue clue){
    //Variables
    string answer; //Player answer
    int counter=0; //Amount of empty letters in keyboard array
    int score=30;  //Points=score*counter
    //Input Data
    categry(clue.categry);
    viewBrd(phrase,key,strlen(clue.phrase),alpha,a);
    cout<<"Input the final answer: ";
    getline(cin,answer);
    //Process Data
    for(int i=0;i<strlen(clue.phrase);i++){ //Convert to uppercase
        answer[i]=static_cast<char>(toupper(answer[i]));
    }
    for(int i=0;i<strlen(clue.phrase);i++){
        if(phrase[i].letter!=answer[i]){//Check to see if all letters match
            return false;               //Phrase did not match answer
        }
    }
    for(int i=0;i<strlen(clue.phrase);i++){ //Go through phrase array to add
        if(phrase[i].isUsed==false){ //points for each letter that is not used
            counter++;
        }
    }
    //Output Data
    score*=counter;
    cout<<"You gain 30 points for each hidden letter you guessed"<<endl;
    cout<<"You gain "<<score<<" points"<<endl;
    a.score+=score;

    return true;
}

void lderBrd(){
    //Variables
    fstream in;         //Input from file
    int n;              //Size of string read from file
    vector<Player> user;//Array of User structures
    Player temp;        //Temp Player to swap for
    int i=0;            //Size of array
    bool flag;
    string a;           //Player inputs to continue
    //Open files
    in.open("users.dat",ios::in|ios::binary);
    if(in.fail()){
        cout<<"users.dat not found"<<endl;
        return;
    }
    //Input Data
    while(in.read(reinterpret_cast<char *>(&n),sizeof(n))){ //Get size of string
        temp.name.resize(n);     //Resize string size by size
        in.read(&temp.name[0],n);//In name and score
        in.read(reinterpret_cast<char *>(&temp.score),sizeof(unsigned int));
        user.push_back(temp);    //Push back array by one at a time
        i++;
    }
    //Process Data
    do{
        flag=false;;
        for(int j=0;(j<i-1);j++){
            if(user[j].score<user[j+1].score){ //Swap greatest to least
                temp=user[j+1];
                user[j+1]=user[j];
                user[j]=temp;
                flag=true;
            }
        }
    }while(flag==true);
    //Output Data
    cout<<"Sorted Leaderboard:"<<endl;
    for(int j=0;j<i;j++){
        cout<<user[j].name<<endl;
        cout<<setw(5)<<right<<user[j].score<<" points"<<endl<<endl;
    }
    cout<<"Press enter to continue";
    getline(cin,a);
}
```