Project 2

Word Guessing Game

CIS-7 43732
Hodnett, Victoria
June 9, 2014

Introduction

A simple word guessing game is a good way to pass the time, whether it is in the form of hangman, the wheel of fortune, or simply being given the length of the word to be guessed.

In the case where no prior information is given, only the length of the word, you initially depend on your luck per say to get the ball rolling, and from then on strategize to think of a valid word, so that you might fill in the remaining spaces.

Game Play

At the start of the game, the user will be prompted to login with a username and password or to create a username and password before continuing.

From that point, the user must enter in the size of the word they would like to guess. If there aren't any words that size in the word list, the user will be prompted to enter in another number until that size is found. Once the size is found, the user will then be presented with a percentage chance of them being able to guess the word in one try (it will be extremely low. Good luck!).

Next, the user must start off by entering in a letter to guess. With their submission, they will either be presented a statement that says that letter is definitely not in the word, after which they must enter in a new letter to guess, or a statement that says it might be in the word. Receiving the latter statement, the user must then guess the *position* of the letter – zero to one number less than the size of the word. The user will be given yet another probability – the chance they will be able to guess the position in one try (this is the *lowest* probability, considering some words may have repeated letters). If the user guesses all the positions and the letter is not found in the word (keep in mind, the indication was that it *might* have been in the word), then they are told so and must guess another letter. If the user guesses the correct position of the letter, the letter replaces the number in that position and they are then prompted to guess another letter.

This process continues until all letters are guessed and the secret word is presented.

Summary

Lines of Code	218
Comment Lines	32
Blank Lines	26
Total Lines	276

Completing the entire project took about a week. Overall I didn't really have any issues developing this project; it was very easy to implement.

In this program I incorporated the use of hashing to protect the usernames and passwords entered for play. I also made use of a simple bloom filter, to aid in protecting the secret word to be guessed, along with a simple form of encryption to place the secret word in a solution array.

I also decided to make use of probabilities and presented the user with the probability of being able to guess the word in one single try, along with the probability of being able to guess a particular letter's position in one try.

Pseudocode

main()

CALL login

IF login() == true THEN

    REPEAT

        WRITE enter number size of word to guess

        READ number

        CLEAR sbset

        CALL subsets with sbset, number and setfound

```
        UNTIL setfound==true
        CALL fillBitAry with a and word
        FOR i=0 to length of word
                ASSIGN i+'0' to guess[i]
        ENDFOR
        FOR i=0 to length of word
                CALL encrypt with word[i]
                ASSIGN result from encrypt to solution[i]
        ENDFOR
        WRITE length of word to guess
        CALL guessing with a, guess, solution, and word.length()
ELSE
        WRITE invalid user name or password
ENDIF
EXIT program
```

login()

```
WRITE enter 1 to login or 2 to create an account
READ choice
IF choice == '1' THEN
        OPEN Data.txt for input
        ASSIGN 0 to j
        WHILE(READ nums[j++] from Data.txt)
        ENDWHILE
        CLOSE Data.txt
        WRITE enter username
```

```
        READ username
        WRITE enter password
        READ password
        CALL BKDRHash with userName
        ASSIGN BKDRHash to hashUser
        CALL BKDRHash with password
        ASSIGN BKDRHash to hashPW
        ASSIGN 0 to found
        FOR i=0 to j-1
                IF hashUser==nums[i] THEN
                        INCREMENT found
                        ASSIGN i to indx
                ENDIF
        ENDFOR
        IF found>0 THEN
                IF hashPW==nums[indx+1] THEN
                        return true
                ELSE
                        return false
                ENDIF
        ELSE
                return false
        ENDIF
ELSE IF choice == '2'
        OPEN Data.txt for output
```

```
        WRITE enter username
        READ username
        WRITE enter password
        READ password
        CALL BKDRHash with userName
        ASSIGN BKDRHash to hashUser
        CALL BKDRHash with password
        ASSIGN BKDRHash to hashPW
        WRITE hashUser and hashPW to Data.txt
        CLOSE Data.txt
        return true
ELSE
        return false
ENDIF
```

subsets()

```
OPEN wordlist.txt
WHILE infile >> temp
        file.push_back(temp)
ENDWHILE
FOR i=0 to size of file
        IF file[i].length()==num
                words.push_back(file[i])
        ENDIF
ENDFOR
IF words.size()==0
```

```
        WRITE set is empty
        ASSIGN false to found
ELSE
        ASSIGN true to found
ENDIF
```

choose()

```
ASSIGN rand()%words.size() to indx
RETURN words[indx]
```

fillBitAry()

```
FOR i=0 to length of w
        ADD w[i] to temp
        CALL findIndx with temp and 1
        ASSIGN result of findIndx to temp1
        CALL findIndx with temp and 2
        ASSIGN result of findIndx to temp2
        ASSIGN 1 to a[temp1]
        ASSIGN 1 to a[temp2]
ENDFOR
```

searchBitAry()

```
ADD ltr to temp
CALL findIndx with temp and 1
ASSIGN result of findIndx to temp1
CALL findIndx with temp and 2
ASSIGN result of findIndx to temp2
IF a[temp1]==1 OR a[temp2]==1
```

```
        WRITE letter is possibly in word
        RETURN true
ELSE
        WRITE letter is not in word
        RETURN false
ENDIF
```

findIndx()

```
IF hash==1
        CALL APHash with a
        ASSIGN result of APHash to index
        RETURN index%52
ELSE
        CALL PJWHash with a
        ASSIGN result of PJWHash to index
        RETURN index%52
ENDIF
```

encrypt()

```
ASSIGN a – ‘0’ to temp
ADD 10 to temp
RETURN temp + ‘0’
```

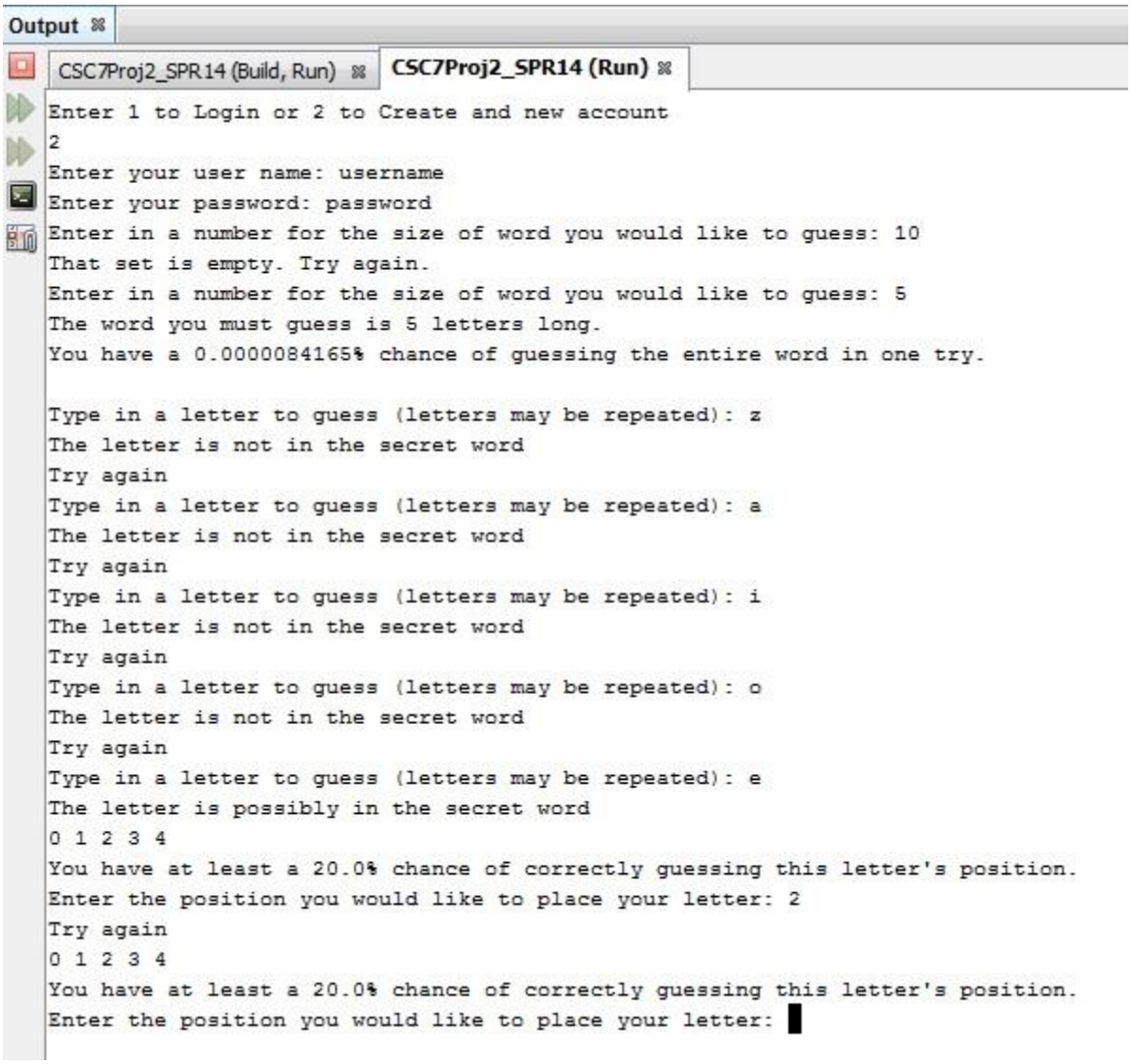
guessing()

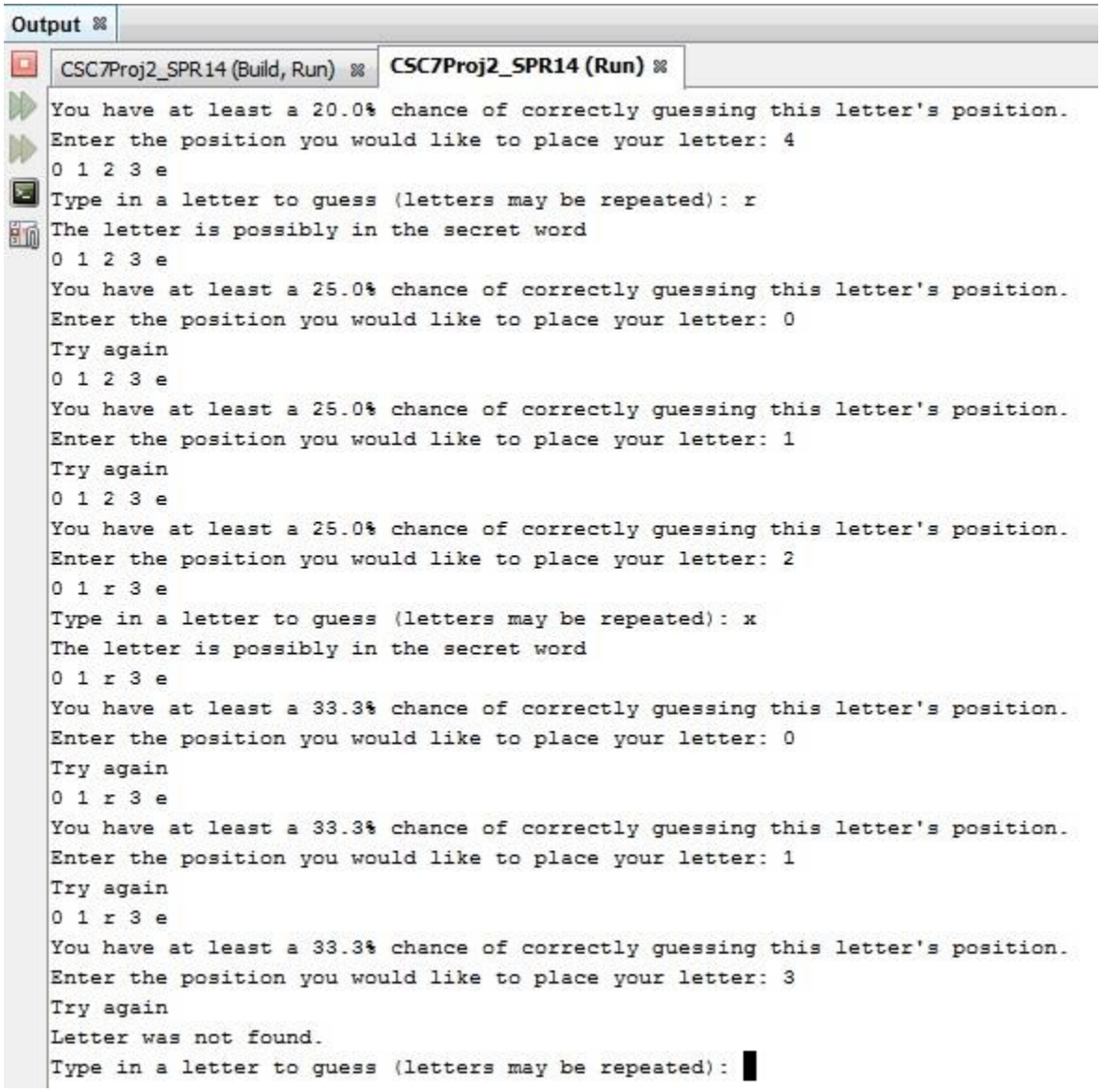
```
ASSIGN 0 to filled
WRITE percentage chance of guessing word
REPEAT
        ASSIGN 0 to count
```

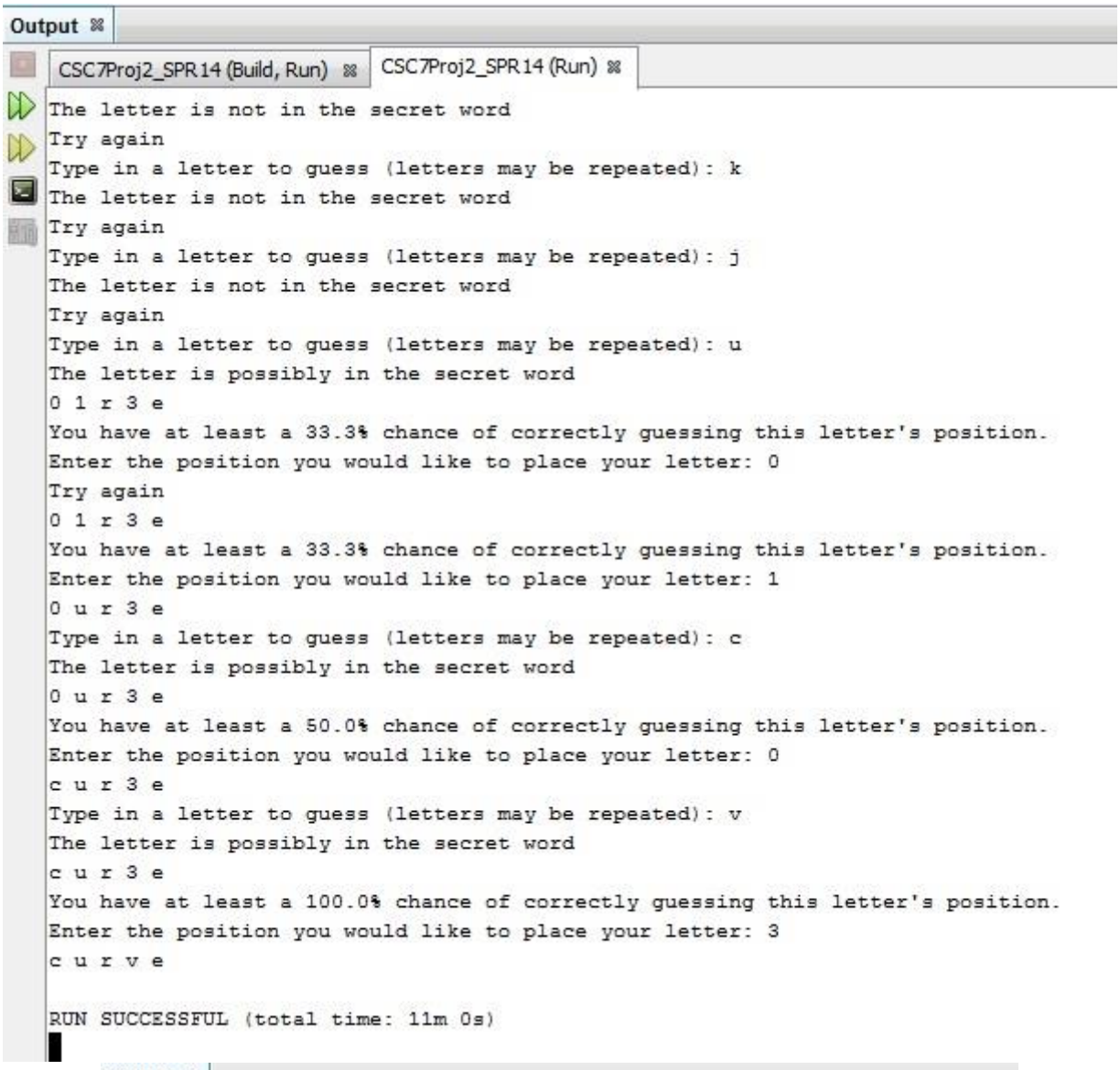
```
ASSIGN false to outerRepeat
WRITE type in letter to guess
READ letter
ASSIGN tolower(letter) to letter
CALL searchBitAry with a and letter
ASSIGN result of searchBitAry to found
IF found=false
        WRITE try again
        ASSIGN true to outerRepeat
ELSE
        REPEAT
                ASSIGN false to innerRepeat
                CALL printPostn with g and n
                WRITE percentage chance of guessing letter
                WRITE enter position to place letter
                READ pos
                CALL encrypt with letter
                IF result of encrypt equals s[pos]
                        ASSIGN letter to g[pos]
                        INCREMENT filled
                        CALL printPOstn with g and n
                        ASSIGN true to outerRepeat
                        ASSIGN false to innerRepeat
                ELSE
                        INCREMENT count
```

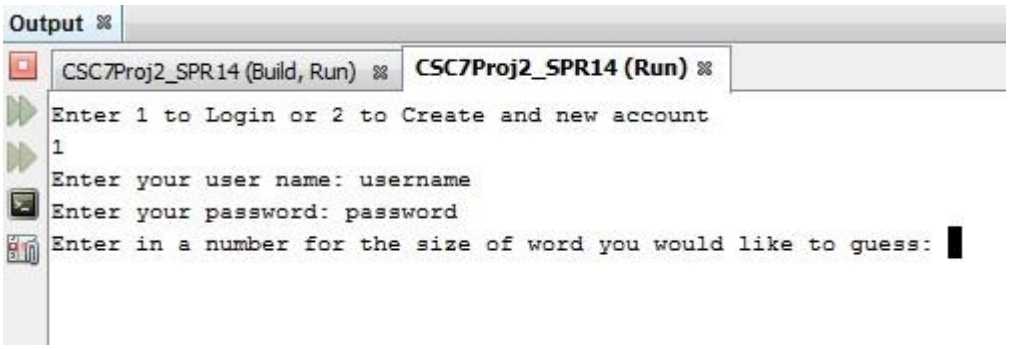
```
                    WRITE try again
                    ASSIGN true to innerRepeat
                    IF count >= (n-filled)
                        WRITE letter not found
                        ASSIGN false to innerRepeat
                        ASSIGN true to outerRepeat
                    ENDIF
                ENDIF
            UNTIL innerRepeat == false
            IF filled < n
                ASSIGN true to outerRepeat
            ELSE
                ASSIGN false to outerRepeat
            ENDIF
        ENDIF
UNTIL outerRepeat == false
printPostn()
FOR i=0 to n
    WRITE a[i]
ENDFOR
```

Sample Input/Output


Output
CSC7Proj2_SPR14 (Build, Run)
CSC7Proj2_SPR14 (Run)
Enter 1 to Login or 2 to Create and new account
2
Enter your user name: username
Enter your password: password
Enter in a number for the size of word you would like to guess: 10
That set is empty. Try again.
Enter in a number for the size of word you would like to guess: 5
The word you must guess is 5 letters long.
You have a 0.0000084165% chance of guessing the entire word in one try.
Type in a letter to guess (letters may be repeated): z
The letter is not in the secret word
Try again
Type in a letter to guess (letters may be repeated): a
The letter is not in the secret word
Try again
Type in a letter to guess (letters may be repeated): i
The letter is not in the secret word
Try again
Type in a letter to guess (letters may be repeated): o
The letter is not in the secret word
Try again
Type in a letter to guess (letters may be repeated): e
The letter is possibly in the secret word
0 1 2 3 4
You have at least a 20.0% chance of correctly guessing this letter's position.
Enter the position you would like to place your letter: 2
Try again
0 1 2 3 4
You have at least a 20.0% chance of correctly guessing this letter's position.
Enter the position you would like to place your letter:


Output
CSC7Proj2_SPR14 (Build, Run)
CSC7Proj2_SPR14 (Run)
You have at least a 20.0% chance of correctly guessing this letter's position.
Enter the position you would like to place your letter: 4
0 1 2 3 e
Type in a letter to guess (letters may be repeated): r
The letter is possibly in the secret word
0 1 2 3 e
You have at least a 25.0% chance of correctly guessing this letter's position.
Enter the position you would like to place your letter: 0
Try again
0 1 2 3 e
You have at least a 25.0% chance of correctly guessing this letter's position.
Enter the position you would like to place your letter: 1
Try again
0 1 2 3 e
You have at least a 25.0% chance of correctly guessing this letter's position.
Enter the position you would like to place your letter: 2
0 1 r 3 e
Type in a letter to guess (letters may be repeated): x
The letter is possibly in the secret word
0 1 r 3 e
You have at least a 33.3% chance of correctly guessing this letter's position.
Enter the position you would like to place your letter: 0
Try again
0 1 r 3 e
You have at least a 33.3% chance of correctly guessing this letter's position.
Enter the position you would like to place your letter: 1
Try again
0 1 r 3 e
You have at least a 33.3% chance of correctly guessing this letter's position.
Enter the position you would like to place your letter: 3
Try again
Letter was not found.
Type in a letter to guess (letters may be repeated):





Data Types Used

Data Type	Example	Description	Location
int	size	Size of bit array	main()
char	choice	Record user's choice for how to proceed	main()
bool	outerRepeat	Determine repetition of outer loop	guessing()
ifstream	infile	Used to read in words	subsets()
fstream	file	Used to read and store usernames and passwords	login()
string	temp	Used to read in words from file	subsets()
unsigned int	nums[25]	Store hashes of usernames and passwords	login()
vector<string>	sbset	To hold subset of words of certain length	Main()
vector<bool>	a	For bit array	Main()

C++ Constructs (Savitch, 8th Edition)

Chapter	Construct	Location
2	Variables	main
	Data types	main
	Input/output	login
	Formatting/iomanip	guessing
	string	login
	if, if-else, if-else-if	login
	do-while/while	main, login
	increment	guessing
3	Boolean expression	main
	For loop	printPostn
4	Type casting	main

	Functions, function calls	main
	Call by value	searchBitAry
5	Void functions	login
	Call by reference	choose
	Functions calling functions	guessing
6	File i/o	login
	Appending file	login
	tolower	guessing
7	Arrays	printPostn

References

The hashing functions I used were downloaded from:

- http://www.partow.net/programming/hashfunctions/

Source Code

```
/*
 * File:   main.cpp
 * Author: Victoria Hodnett
 * Created on June 2, 2014, 10:03 AM
 * CSC7 Project 2
 */

//System Libraries
#include <cstdlib>
#include <iostream>
#include <string>
#include <fstream>
#include <vector>
#include <ctime>
#include <cmath>
#include <iomanip>
using namespace std;

//User-defined Libraries
#include "GeneralHashFunctions.h"

//Global Constants

//Function Prototypes
bool login();
```

```
void subsets(vector<string>&,int,bool&);
string choose(vector<string>&);
void fillBitAry(vector<bool>&,string);
unsigned int findIndx(string,int);
char encrypt(char);
void guessing(vector<bool>&,char[],char[],int);
void printPostn(char[],int);
bool searchBitAry(vector<bool>&,char);

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

    if(login()){
        //Choose word
        int number;
        vector<string>sbset(0);
        bool setfound;
        do{
            cout << "Enter in a number for the size of word you would "
                "like to guess: ";
            cin >> number;
            sbset.clear();
            subsets(sbset,number,setfound);
        }while(!setfound);

        string word = choose(sbset);
        //cout << word << endl;
        //Fill Bit Array
        int size = 52;
        vector<bool>a(size);
        fillBitAry(a,word);
        //Initialize guess and solution arrays
        char guess[word.length()], solution[word.length()];
        for(int i=0;i<word.length();i++){
            guess[i]=i+'0';
        }
        for(int i=0;i<word.length();i++){
            solution[i]=encrypt(word[i]);
        }
        //Make guess
        cout << "The word you must guess is " << word.length()
                << " letters long. " << endl;
        guessing(a,guess,solution,word.length());
        }else{
        cout << "Invalid user name and/or password. Exiting now..." << endl;
    }
    return 0;
}

//Login before playing game
bool login(){
    fstream file;
    char choice;
    unsigned int hashUser,hashPW;
    string userName,password;
    unsigned int nums[25]={};
```

```
    cout << "Enter 1 to Login or 2 to Create and new account" << endl;
    cin >> choice;

    if(choice=='1'){
        file.open("Data.txt", ios::in);
        int j=0;
        while(file >> nums[j++]){
        }
        file.close();
        cout << "Enter your user name: ";
        cin >> userName;
        cout << "Enter your password: ";
        cin >> password;

        hashUser = BKDRHash(userName);
        hashPW = BKDRHash(password);

        int found=0,indx;
        //search for hashUser in inputs
        //if found, check password against the first index after
        //if password matches, return true
        //if not found, return false;
        for(int i=0;i<j-1;i++){
            if(hashUser==nums[i]){
                found++;
                indx=i;
            }
        }
        if(found>0){
            if(hashPW==nums[indx+1])return true;
            else return false;
        }else return false;
    }else if(choice=='2'){
        file.open("Data.txt", ios::out | ios::app);
        //Input account info
        cout << "Enter your user name: ";
        cin >> userName;
        cout << "Enter your password: ";
        cin >> password;

        hashUser = BKDRHash(userName);
        hashPW = BKDRHash(password);

        file << hashUser << endl;
        file << hashPW << endl;
        file.close();
        return true;
    }else{
        return false;
    }
}

//Choose subset of words
void subsets(vector<string>&words, int num, bool&found){
    vector<string>file;
    string temp;
    ifstream infile;
```

```
    infile.open("wordlist.txt");
    while(infile >> temp){
        file.push_back(temp);
    }
    infile.close();

    for(int i=0;i<file.size();i++){
        if(file[i].length()==num)
            words.push_back(file[i]);
    }

    if(words.size()==0){
        cout << "That set is empty. Try again. " << endl;
        found = false;
    }else{
        found = true;
    }
}

//Choose word
string choose(vector<string>& words){
//    //Test input
//    for(int i=0;i<words.size();i++){
//        cout << words[i] << " ";
//    }cout << endl;
    //Choose random word
    int indx = rand()%words.size();
    return words[indx];
}

void fillBitAry(vector<bool>& a,string w){
    for(int i=0;i<w.length();i++){
        string temp = "";
        temp+=w[i];
        unsigned int temp1 = findIndx(temp,1);
        unsigned int temp2 = findIndx(temp,2);

        a[temp1]=1;
        a[temp2]=1;
    }
}

bool searchBitAry(vector<bool>&a,char ltr){
    string temp = "";
    temp+=ltr;
    unsigned int temp1 = findIndx(temp,1);
    unsigned int temp2 = findIndx(temp,2);

    if(a[temp1]==1||a[temp2]==1){
        cout << "The letter is possibly in the secret word" << endl;
        return true;
    }else{
        cout << "The letter is not in the secret word" << endl;
        return false;
    }
}

unsigned int findIndx(string a, int hash){
```

```
    unsigned int index;
    if(hash==1){
        index = APHash(a);
        return index%52;
    }else{
        index = PJWHash(a);
        return index%52;
    }
}

char encrypt(char a){
    int temp = a - '0';
    temp+=10;
    return (temp + '0');
}

void guessing(vector<bool>&a,char g[], char s[],int n){
    bool outerRepeat,innerRepeat;
    int filled = 0;
    cout << "You have a " << fixed << showpoint << setprecision(10)
            << pow(1.0/26,n) * 100 << "% chance of guessing the entire "
            "word in one try. " << endl << endl;
    do{
        char letter;
        int count = 0;
        outerRepeat = false;
        cout << "Type in a letter to guess (letters may be "
                "repeated): ";
        cin >> letter;
        letter=tolower(letter);

        bool found=searchBitAry(a,letter);
        if(!found){
            //Try again
            cout << "Try again" << endl;
            outerRepeat = true;
        }else{
            do{
                innerRepeat = false;
                //Print positions
                printPostn(g,n);
                //Guess position
                int pos;
                cout << "You have at least a " << fixed << showpoint
                        << setprecision(1) << 1.0/(n-filled) * 100 << "% chance of "
                        "correctly guessing this letter's position. " << endl;
                cout << "Enter the position you would like to "
                        "place your letter: ";
                cin >> pos;
                if(encrypt(letter)==s[pos]){
                    g[pos]=letter;
                    filled++;
                    printPostn(g,n);
                    outerRepeat = true;
                    innerRepeat = false;
                }else{
                    count++;
                    cout << "Try again " << endl;
```

```
                    innerRepeat = true;
                    if(count>=(n-filled)){
                        cout << "Letter was not found. " << endl;
                        innerRepeat = false;
                        outerRepeat = true;
                    }
                }
            }while(innerRepeat);
            if(filled<n)outerRepeat = true;
            else outerRepeat = false;
        }
    }while(outerRepeat);
}

void printPostn(char a[],int n){
    for(int i=0;i<n;i++){
        cout << a[i] << " ";
    }cout << endl;
}
```