```
/* CISC 1610
 * Louis Boulas
 * Final Project
 * hangman.cpp
 */


#include <iostream>
#include <string>
#include <cstring>
#include <vector>
#include <bits/stdc++.h>
#include <fstream>
#include <cstdlib>
using namespace std;


//------------------------------FUNCTION DECLARATIONS-----------------------

//Welcome message and displays the rules of the game/point system
void prompt_welcome();

// menu prompt (choose difficulty, easy, medium or hard)
void prompt_diff();

// function that randoms a number 1-5, and returns that value
int rand_num();

// function that sets the secret word to the difficulty choice and sets lives
// accordingly
// reads in difficulty chosen by player
// parameters: difficulty choice, set lives, set secret word, x - randomly
// generated number
string secret_word(int &choice, int &lives, string file_array[][5], int x);

//void function that will display the # of lives the user has and the secret
//word as hyphens/ show the correctly guessed letters;
void display(int lives, char word_hidden[], int length);

// function that returns a char -- the letter that the user inputs
// will prompt for a letter and return it
char prompt_letter();

// function to check if letter is in the secret word or check if letter has
// alrdy guessed
// post conditions: returns true if the letter is not in the word or in wrong
// guesses vector
// returns false if the letter is in either one
bool in_word(string secret_word, char word_hidden[], char letter,
        int length, vector<char>&wrong_guess, int size);

// void function will output all of the letters guessed that were not found
// in secret word
void wrong_choices(vector<char>&wrong_guess, int size);

// bool function that will check if the user has won. will check each char of
// secret word with the letters user inputed.
bool win(string secret_word, char word_hidden[], int length);

// bool function that will read in whether the user wants to try again after
// returns true if they want to try again, returns false if not
bool more_lives(int &lives, int diff_choice, bool &play, string word);

// void function that couts the scoreboard, takes in several parameters that
// keeps track of player lives, total attempts, # of words guessed,
// total points
void scoreboard(int attempts, int unused_lives, int words_guessed, int points);

//void function that asks user if they want to play again with a new word
//post condition: will either send player to difficulty choice function or
//quit the game.
void new_word(bool &lets_play, vector<char>&wrong_guess);

//function that will read in the secret words from a txt file and store them
//into an array
//parameters: 2d string array that will have 3 rows for each difficulty, and
//5 columns for 5 words.
void open_file(string file_array[3][5]);
//---------------------------------CONSTANTS-------------------------------



int main()
{

    // Variables for the hangman game
```

```
int diff_choice, length, lives, size, x;
string word;
char letter;
bool not_in_word, wp(false), lets_play(true);
vector<char>wrong_guess;

// Variables for to keep track of player score
int words_guessed(0), points(0), used_lives(0), attempts(0);

prompt_welcome();

// read in all secret words from file and store into an arr
string hangman_array[3][5];
open_file(hangman_array);
cout << hangman_array[1][2] << endl;



while (lets_play)
{

    prompt_diff();

    // initialize difficulty, initialize lives

    x = rand_num();

    word = secret_word(diff_choice, lives, hangman_array, x);

    length = word.length();

    // making an array for hidden word; size of array = length.

    char word_hidden[length];

    bool run(true), play(true);


    while (run)
    {
        // store "-" in array (players array)
        for (int i=0; i < length; ++i)
        {
            word_hidden[i] = '-';
        }


        while (play)
        {
            display(lives, word_hidden, length);

            // if no wrong letters dont display vector of wrong letters
            size = wrong_guess.size();

            if (size !=0)
                wrong_choices(wrong_guess, size);

            // ask user for a letter
            letter = prompt_letter();

            // check if letter in the secret word
            not_in_word = in_word(word, word_hidden, letter, length,
                    wrong_guess, size);

            if (not_in_word)
            {
                --lives;
                ++used_lives;
                wrong_guess.push_back(letter);
            }

            // wp short for well played -- check if user won
            wp = win(word, word_hidden, length);

            if (wp)
            {
                cout << word << "\n\nWoohooooo! You got it!!" << endl;
                attempts++;
                points += diff_choice;
                points += lives;
                words_guessed++;
                play = false;
                run = false;
                break;
            }

            // check if user has run out of lives
```

```
                if (lives == 0)
                {
                    points--;
                    attempts++;
                    cout << "You ran out of lives!\n"
                         << "Would you like to keep trying and have your lives "
                         << "reset? [Y]es, anything else will be no.";
                    run = more_lives(lives, diff_choice, play, word);
                    cout << endl;
                }
            }
        }

        // display scoreboard
        scoreboard(attempts, used_lives, words_guessed, points);

        // ask user if they want to play again and reset vector of wrong
        // letters
        new_word(lets_play, wrong_guess);

    }



    return 0;
}




//-----------------------------FUNCTION DEFINITIONS-------------------------
void prompt_welcome()
{
    cout << "Welcome to the Hangman Game!\n\n"
         << "We keep track of score here! So go get them points for every"
         << " word correctly guessed!\n"
         << "1, 2 or 3 points are awarded based on the difficulty chosen.\n"
         << "You will receive an extra point for every life you have leftover!"
         << "\nCareful, if you run out of lives, you lose a point!\n\n"
         << "Goodluck!" << endl;
}

void open_file(string file_array[3][5])
{
    ifstream in_file;
    in_file.open("secret_words.txt");

    if (!in_file)
    {
        cout << "unable to open file";
        exit(1);
    }
    else
    {
        for (int i=0; i<3; ++i)
        {
            for (int j=0; j<5; ++j)
            {
                in_file >> file_array[i][j];
            }
        }
    }
    in_file.close();
}

int rand_num()
{
    int num;
    srand (time(NULL));
    num = rand() % 5;
    --num;
    return num;
}

void prompt_diff()
{
    cout << "Please choose a level of difficulty:\n"
         << "[1] \t Easy (5 or less letters)\n"
         << "[2] \t Medium (6-11 letters)\n"
         << "[3] \t Hard (12+ letters)\n";
}

string secret_word(int &choice, int &lives, string file_array[][5], int x)
{
    string word;
    bool wrong_input(true);
```

```
    while (wrong_input)
    {
        cin >> choice;

        switch (choice)
        {
            case 1:
                word = file_array[0][x];
                lives = 6;
                wrong_input = false;
                break;
            case 2:
                word = file_array[1][x];
                lives = 8;
                wrong_input = false;
                break;
            case 3:
                word = file_array[2][x];
                lives = 10;
                wrong_input = false;
                break;
            default:
                break;
        }
    }

    cout << endl << endl;
    return word;
}

void display(int lives, char word_hidden[], int length)
{
    for (int i =0; i < length; ++i)
    {
        cout << word_hidden[i];


    }
    cout << "\nLife count: " << lives << endl;
}

char prompt_letter()
{
    char letter;
    cout << "Enter a letter: ";
    cin >> letter;
    cout << endl;
    return letter;
}

bool in_word(string secret_word, char word_hidden[], char letter,
        int length, vector<char>&wrong_guess, int size)
{
     bool wrong_letter(true);

     for (int i=0; i < length; ++i)
     {
         if (letter == secret_word[i])
         {
             word_hidden[i] = secret_word[i];
             wrong_letter = false;
         }
     }

     for (int i=0; i<size; ++i)
     {
         if (letter == wrong_guess[i])
         wrong_letter = false;
     }

     return wrong_letter;
}

void wrong_choices(vector<char>&wrong_guess, int size)
{
    cout << "Letters not in secret word: ";
    for (int i=0; i<size; ++i)
        cout << wrong_guess.at(i);
    cout << endl;
}

bool win(string secret_word, char word_hidden[], int length)
{
    bool winner(true);

    for (int i=0; i<length; ++i)
```

```
    {
        if (secret_word[i] != word_hidden[i])
        {
            winner = false;
            break;
        }
    }
    return winner;
}

bool more_lives(int &lives, int diff_choice, bool &play, string word)
{
    char more_lives;
    bool go;

    cin >> more_lives;

    if (more_lives == 'Y' || more_lives == 'y')
    {
        go = true;
        switch (diff_choice)
        {
            case 1:
                lives = 6;
                break;
            case 2:
                lives = 8;
                break;
            case 3:
                lives = 10;
        }
    }
    else
    {
        cout << "The secret word was: " << word << endl;
        go = false;
        play = false;
    }
    return go;
}

void scoreboard(int attempts, int used_lives, int words_guessed, int points)
{
    cout << "\n\nSCOREBOARD:\n";
    cout << "Total Attempt(s): " << attempts << endl;
    cout << "Lives Expended: " << used_lives << endl;
    cout << "# of Word(s) Guessed: " << words_guessed << endl;
    cout << "Total Score: " << points << endl << endl;
}

void new_word(bool &lets_play, vector<char>&wrong_guess)
{
    char choice;

    cout << "Type 'Y' if you would like to play again with a new word.\n"
         << "Anything else will quit the game.";

    cin >> choice;

    if (choice == 'Y' || choice == 'y')
    {
        lets_play = true;
        wrong_guess.clear();
    }
    else
    {
        cout << "Thanks for play the Hangman Game!\n";
        lets_play = false;
    }
}
```