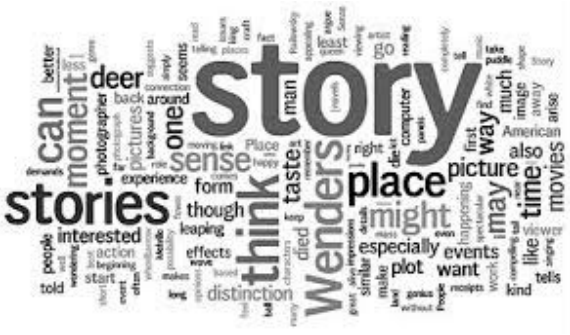
CPP Project



Presenting by:

Niv Toren

Yossef Sohil Parizade

Chapter 1:

A run file is required for each of the questions separately, in which all the functions you were asked to write are defined.

Exercises 1.1: Write a program that reads words.txt and prints only the words with more than 20 characters (not counting whitespace).

Answer:

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main() {
    std::ifstream fin("words.txt");
    string myText;
    while (fin >> myText)
    {
        if (myText.length() > 20)
        {
            cout << myText << endl;
        }
    }
    return 0;
}
```

Exercises 1.2: In 1939 Ernest Vincent Wright published a 50,000-word novel called Gadsby that does not contain the letter “e”. Since “e” is the most common letter in English, that’s not easy to do. In fact, it is difficult to construct a solitary thought without using that most common symbol. It is slow going at first, but with caution and hours of training you can gradually gain facility. All right, I’ll stop now. Write a function called has_no_e that returns true if the given word doesn’t have the letter “e” in it. Modify your program from the previous section to print only the words that have no “e” and compute the percentage of the words in the list that have no “e”.

Answer:

```
#include <iostream>
#include <fstream>
#include <string>


using namespace std;

bool has_no_e(string word)
{
    for (int i = 0; i < word.length(); i++)
    {
        if (word[i] == 'e')
            return false;
    }
    return true;
}

int main()
{
    std::ifstream fin("words.txt");
    string myText;
    int count = 0;
    int num_of_words = 113783;

    while (fin >> myText)
    {
        if (has_no_e(myText))
        {
            count++;
            cout << myText << endl;
        }
    }


    cout << (double(count) / double(num_of_words)) * 100 << "%" << endl;

    return 0;

}
```

Exercise 1.3: Write a function named avoids that takes a word and a string of forbidden letters, and that returns true if the word doesn't use any of the forbidden letters. Modify your program to prompt the user to enter a string of forbidden letters and then print the number of words that don't contain any of them. Can you find a combination of five forbidden letters that excludes the smallest number of words?

Answer:

```
#include <iostream>
#include <fstream>
#include <string>

using namespace std;

bool avoid(string word, string forbidden)
{
    for (int i = 0; i < word.length(); i++)
    {
        for (int j = 0; j < forbidden.length(); j++)
        {
            if (word[i] == forbidden[j])
            {
                return false;
            }
        }
    }
    return true;
}

int main()
{
    std::ifstream file("words.txt");
    string word;
    string forbidden;
    int count = 0;

    cout << "Type forbidden letters: ";
    cin >> forbidden;

    while (file >> word)
    {
        if (avoid(word, forbidden) == true)
        {
            count++;
        }
    }

    cout << count << endl;

    return 0;
}
```

Exercise 1.4: Write a function named uses_only that takes a word and a string of letters, and that returns true if the word contains only letters in the list. Can you make a sentence using only the letters acefhlo? Other than “Hoe alfalfa”?

Answer:

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>

bool uses_only(const std::string& word, const std::string& allowed)
{
    for (char c : word)
        if (allowed.find(c) == std::string::npos)
            return false;
    return true;
}

int main()
{
    std::string word;
    std::string allowed = "acefhlo";
    std::ifstream file("words.txt");

    std::vector<std::string> words_made_up_of_letters;

    if (file.is_open())
    {
        while (file >> word)
        {
            if (uses_only(word, allowed))
                words_made_up_of_letters.push_back(word);
        }
        file.close();
    }
    else
    {
        std::cerr << "Unable to open file" << std::endl;
        return 1;
    }

    std::cout << "This words using only the letters acefhlo:" << std::endl;
    for (const std::string& word : words_made_up_of_letters)
        std::cout << word << std::endl;

    std::cout << "The sentence I made using acefhlo letters is: " << std::endl;
    std::cout << "Each leaf has an echo of its own, oh oh" << std::endl;

    return 0;
}
```

Exercise 1.5: Write a function named uses_all that takes a word and a string of required letters, and that returns true if the word uses all the required letters at least once. How many words are there that use all the vowels aeiou? How about aeiouy?

Answer:

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>

using namespace std;

bool uses_all(string word, string string) {
    int count = 0;
    for (int i = 0; i < string.length(); i++) {
        if (word.find(string[i]) != string::npos) {
            count++;
        }
    }
    if (count == string.length()) {
        return true;
    }
    return false;
}

int find_uses_all_vowels(vector<string> list, string vowels) {
    int count = 0;
    for (int i = 0; i < list.size(); i++) {
        if (uses_all(list[i], vowels)) {
            count++;
        }
    }
    return count;
}

int main() {
    ifstream file("words.txt");
    if (!file.is_open()) {
        cerr << "Error opening file" << endl;
        return 1;
    }

    vector<string> word_list;
    string word;
    while (file >> word) {
        word_list.push_back(word);
    }
    file.close();

    cout << "Words using all vowels 'aeiou': " << find_uses_all_vowels(word_list,
"aeiou") << endl;
    cout << "Words using all vowels 'aeiouy': " << find_uses_all_vowels(word_list,
"aeiouy") << endl;
    return 0;
}
```

2.2 Wordle game:

```
#include<iostream>
#include<fstream>
#include<array>
#include<vector>
#include<map>
#include<string>
#include<stdlib.h>
#include<time.h>

const unsigned int length = 5, chances = 5;

enum LetterIs {
	Match,
	Contained,
	Nonexistent
};
std::map<LetterIs, std::string> displayLetterIs{
	{ Match, "Match " },
	{ Contained, "Contained " },
	{ Nonexistent, "Nonexistent " }
};

std::vector<std::string> ExtractText(std::string filePath) {
	std::ifstream file; file.open(filePath);
	std::vector<std::string> text;
	if (file.is_open()) {
		while (!file.eof()) {
			std::string line;
			getline(file, line);
			if (line.length() == length) {
				text.push_back(line);
			}
		}
		file.close();
	}
	else {
		std::cout << "We weren't able to open the file\n";
		exit(1);
	}
	return text;
}

int main() {
	while (true) {
		std::cout << "Wordle Game - Please find the word with 5 letters\n";
		srand((unsigned)time(NULL));
		std::vector<std::string> text = ExtractText("words.txt");
		std::string wordle = text[rand() % text.size()];
		for (int i = 0; i < chances; i++) {
			std::cout << "Guess [" << i + 1 << '/' << chances << "]: ";
			std::string guess; std::cin >> guess;
			if (guess.length() != length) {
				std::cout << "Please type 5 letters" << std::endl;
				continue;
			}
			std::array<LetterIs, length> lettersAre;
```

```
                    for (int j = 0; j < length; j++) {
                          if (guess[j] == wordle[j]) lettersAre[j] = Match;
                          else {
                                bool nonexistent = true;
                                for (int k = 0; k < length; k++) if (guess[j] ==
wordle[k]) {
                                      lettersAre[j] = Contained;
                                      nonexistent = false;
                                      break;
                                }
                                if (nonexistent) lettersAre[j] = Nonexistent;
                          }
                    }
                    std::cout << "Result: ";
                    for (int j = 0; j < length; j++)
                          std::cout << displayLetterIs[lettersAre[j]] << ' ';
                    std::cout << std::endl;
                    if (guess == wordle) {
                          std::cout << "Congratulations, you won the game!\n";
                          break;
                    }
                    if (i == chances - 1) {
                          std::cout << "You lost the game, the word was: " << wordle
<< std::endl;
                    }
              }
              std::cout << "Do you want to play again [y/n]: ";
              char playAgain; std::cin >> playAgain;
              if (playAgain == 'n') break;
      }
      return 0;
}
```

Exercise 3.1: Create now your own image. The most beautiful image may earn some extra points (bonus).
(You can make more than one image, but not more than 3).

Answer:

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
	ofstream image; // what about ifstream image("image.ppm")
	image.open("image1.ppm");
	if (image.is_open()) {
		//place header information
		image << "P3" << endl; //format
		image << "250 250 " << endl; //image size
		image << "255" << endl; //max pixel value
	}
	for (int y = 0; y < 250; y++) {
		for (int x = 0; x < 250; x++) {
			// write values to the x,y pixel
			image << x % 76 << " " << y % 212 << " " << (x + y) % 255 <<
endl;
		}
	}
	image.close();
	return 0;
}
```

(All the pictures' codes contain in the cpp code documents).

Exercise 3.2: create a white circle on a black background like the following image:

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
	ofstream image;
	image.open("dots.ppm");
	if (image.is_open())
	{
		//place header information
		image << "P3" << endl; //format
		image << "250 250 " << endl; //image size
		image << "255" << endl; //max pixel value

		//center of the circle
		int centerX = 36, centerY = 82;
		int radius = 7;

		for (int y = 0; y < 250; y++)
		{
			for (int x = 0; x < 250; x++)
			{
				// check if the current pixel is inside the circle
				if ((x - centerX) * (x - centerX) + (y - centerY) * (y -
centerY) <= radius * radius)
				{
					image << "255 255 255" << endl; //White color
				}
				else
				{
					image << "0 0 0" << endl; //Black color
				}
			}
		}

		image.close();
		return 0;
	}
}
```

Chapter 2:

Question 1.3– Can you find a combination of five forbidden letters that excludes the smallest number of words?
Answer- No, there are so many words.

Question 1.4– Can you make a sentence using only the letters acefhlo? Other than “Hoe alfalfa”?
Answer: Yes, we make a sentence below:

Sentence: Each leaf has an echo of its own, oh oh

Question 1.5- Write a function named `uses_all` that takes a word and a string of required letters, and that returns true if the word uses all the required letters at least once. How many words are there that use all the vowels aeiou? How about aeiouy?
Answer:
Words using all vowels 'aeiou': 598
Words using all vowels 'aeiouy': 42

Wordle game- screenshot:

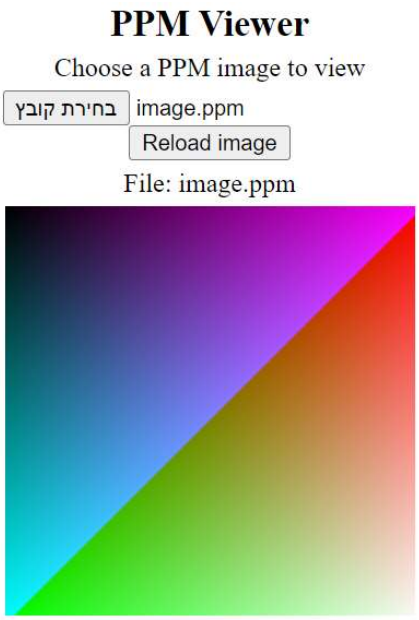
<u>Question 2.2</u>- Wordle game

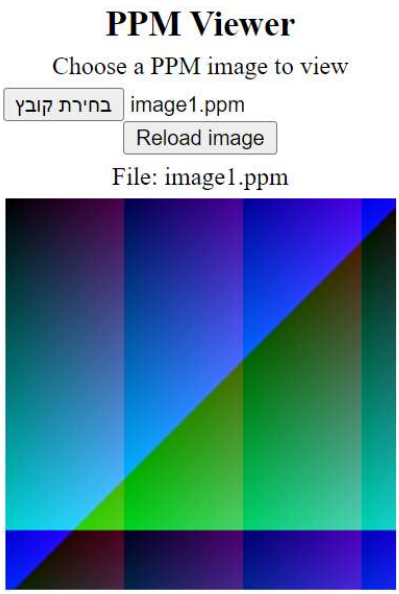
```
C:\Users\ניב טורן\source\repos\cpp_project\x64\Debug\cpp_project.exe
Wordle Game - Please find the word with 5 letters
Guess [1/5]: hello
Result: Nonexistent  Contained  Nonexistent  Nonexistent  Nonexistent
Guess [2/5]: wordle
Please type 5 letters
Guess [3/5]: const
Result: Nonexistent  Nonexistent  Nonexistent  Contained  Nonexistent
Guess [4/5]: toxic
Result: Nonexistent  Nonexistent  Nonexistent  Nonexistent  Nonexistent
Guess [5/5]: debug
Result: Nonexistent  Contained  Nonexistent  Nonexistent  Contained
You lost the game, the word was: gazes
Do you want to play again [y/n]:
```

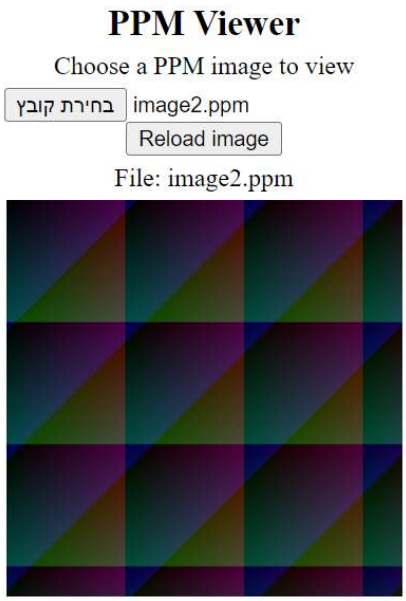
<u>Question 2.3</u>-Can you beat the game?

Answer: We doesn’t win the game.

<u>Question 3.1 pictures -</u> Create now your own image.
Answer:







Also in 3.1 question we ask about x and y location, and the answer is:

In 2D computer graphics and image processing, the X-axis typically runs horizontally, increasing from left to right, and the Y-axis runs vertically, increasing from top to bottom. This is known as a Cartesian coordinate system and is commonly used in image and display coordinate systems, with the origin (0,0) usually at the top-left corner of the image.

However, it's worth noting that the exact orientation of the X and Y axes can vary depending on the application, and some systems may use different conventions, such as having the origin at the bottom-left corner, or in the center of the image.

<u>Question 3.2:</u> create a white circle on a black background.

This is what we made: