Assignment 1 – Report

In this assignment an application is made for displaying a school report. A report contains courses (like "Programming 2") and each course has a theory grade and a practical grade.

class Course and enum PracticalGrade

a) Create a `class Course` with the fields:
- Name (name of the course: string)
- Grade (theory grade: int)
- Practical (practical grade: enumeration)

The practical grade can have one the following values: None, Absent, Insufficient, Sufficient and Good.
→ Create an `enum PracticalGrade` for this.

b) Create the following methods to read/display a PracticalGrade and to read/display a Course:

```
PracticalGrade ReadPracticalGrade(string question)
void DisplayPracticalGrade(PracticalGrade practical)
Course ReadCourse(string question)
void DisplayCourse(Course course)
```

→ Use the ReadInt and ReadString methods we've created in week 1 (copy these 3 methods).
→ Check the Read/Display methods by calling them from the `Start` method.

c) In the Start method declare a list of courses:

```
List<Course> report;
```

→ Fill the report with 3 courses. Create a method `List<Course> ReadReport(int nrOfCourses)`.
→ Display the report on screen. Create a method `void DisplayReport(List<Course> report)`.

d) Add the following methods to `class Course`:

```
public bool Passed()
public bool CumLaude()
```

A course is passed if the grade is 55 or higher <u>and</u> the practical grade is Sufficient or Good.
A course is passed Cum Laude if the grade is 80 or higher <u>and</u> the practical grade is Good.

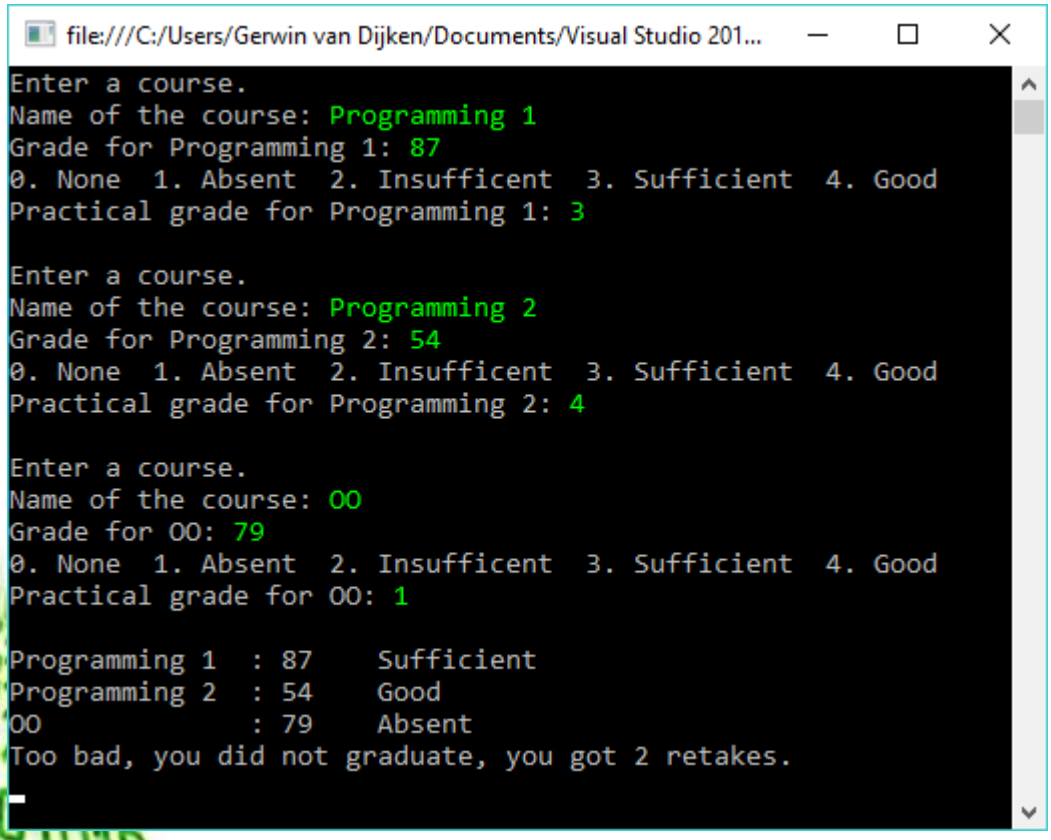
e) Now add code to method `DisplayReport` to determine if the student has graduated or not. If the student did not graduate then display the number of retakes. Also display if the student graduated Cum Laude.

ReadCourse

ReadCourse

ReadCourse

DisplayReport



Assignment 2 – Hangman

class HangmanGame

The purpose of the game Hangman is to guess a secret word. Two essential parts of the game are the secret word and the guessed word so far (containing the guessed letters). See www.playhangman.com.

a) For the game Hangman we create a `class HangmanGame`, containing two strings: `secretWord` and `guessedWord`.
Add to `class HangmanGame` a method with signature:

```
public void Init(string secretWord)
```

This method stores the secret word, and fills the guessed word with dots, as many dots as the number of characters in the secret word.
So, if the secret word is `"backdoor"` then the guessed word will be filled with 8 dots: `"........"`.
Later (in this assignment) we shall add more methods to the class HangmanGame.
→ Test your class with the following code (in the `Start` method):

```
HangmanGame hangman = new HangmanGame();
hangman.Init("backdoor");
Console.WriteLine("The secret word is: " + hangman.secretWord);
Console.WriteLine("The guessed word is: " + hangman.guessedWord);
```

Generating a random word

b) Implement a (Program) method with signature:

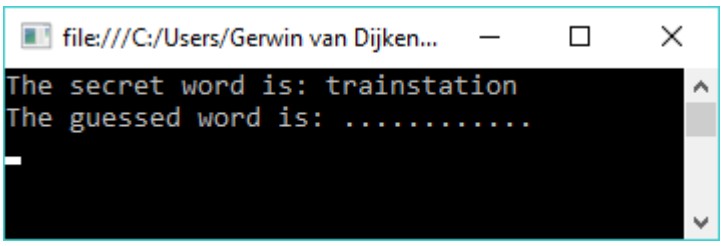
```
List<string> ListOfWords()
```

This method returns a list of 20 (hardcoded) words. One of these words will become the secret word for the hangman game.
→ Declare in the `Start` method a `List<string> words` and fill this list via method `ListOfWords`.

c) Implement a (Program) method with signature:

```
string SelectWord(List<string> words)
```

This method chooses a random word from the list of words.
→ Call this method `SelectWord` from the `Start` method and pass (as a parameter) the list of words. The selected word will serve as secret word.
→ Display (as a test) the selected word in the `Start` method by using the given code (see border above) again. Of course the user must not see this word...



Reading and displaying letters

d) Implement a (Program) method with signature:

```
bool PlayHangman(HangmanGame hangman)
```

This method returns `true` if the secret word has been guessed by the user, otherwise it returns `false`.
To be able to remember the entered letters we'll create a list of char (in method `PlayHangman`):

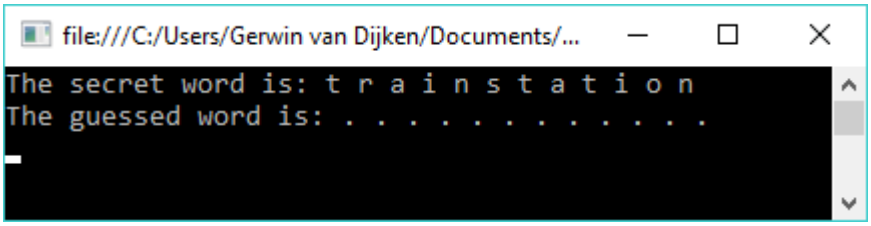
```
List<char> enteredLetters;
```

For now, this method returns `true`.
→ Call the method `PlayHangman` from the `Start`.
For now we will use method `PlayHangman` to test other methods. At the end of the assignment we will finish the implementation of this method.

e) Create a (Program) method with signature:

```
void DisplayWord(string word)
```

This method displays the given word with spaces between the letters. We will use this method everytime we need to display the guessed word.
→ Call method `DisplayWord` from the method `PlayHangman` and test the method (for now) by printing the secret word and the guessed word.



f) Create a (Program) method:

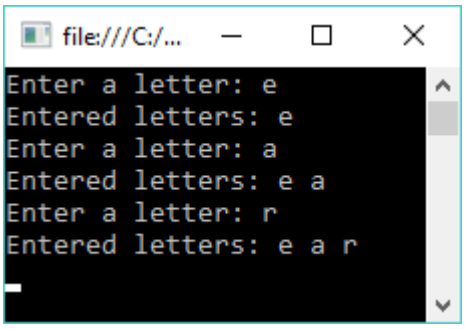
```
void DisplayLetters(List<char> letters)
```

This method displays the letters seperated by spaces.
→ Call the method `DisplayLetters` from the method `PlayHangman` to test the method, by adding some dummy letters to the list of entered letters.

g) Create a (Program) method:

```
char ReadLetter(List<char> blacklistLetters)
```

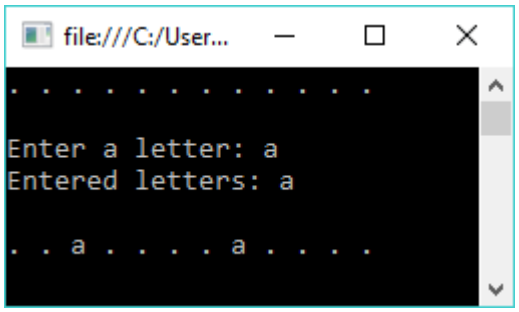
This method reads a letter until this letter is not in the blacklist. To check this use `blacklistLetters.Contains(letter)`.
→ Call the method ReadLetter from the method `PlayHangman` and add the returned letter to the list with entered letters.



h) Add to `class HangmanGame` a method:

```
public bool GuessLetter(char letter)
```

This method checks if the secret word contains the given letter, and returns `true` if it does, `false` otherwise.
If the secret word contains the letter, then change the guessed word by putting this letter in the correct place(s).
→ Test the method by calling it from PlayHangman.



Guessing the word

The user has (max) 8 attempts to guess the hangman word. The count of remaining attempts is decreased everytime the user enters a letter that is not present in the hangman word. The program stops if the user has guessed the word, or if the user has no more attempts left.

i) Add to `class HangmanGame` a method:

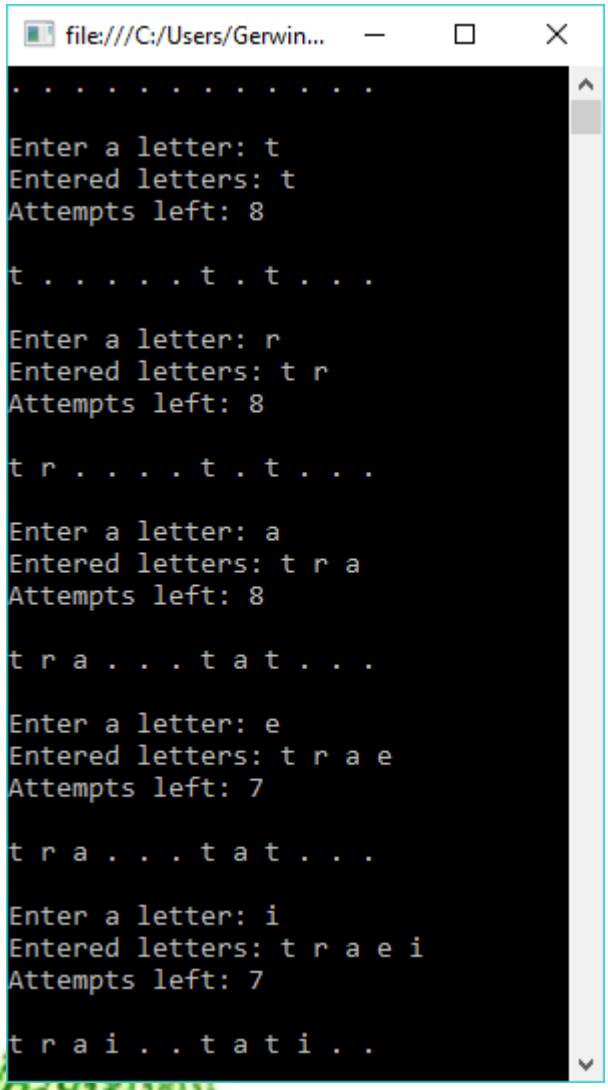
```
public bool IsGuessed()
```

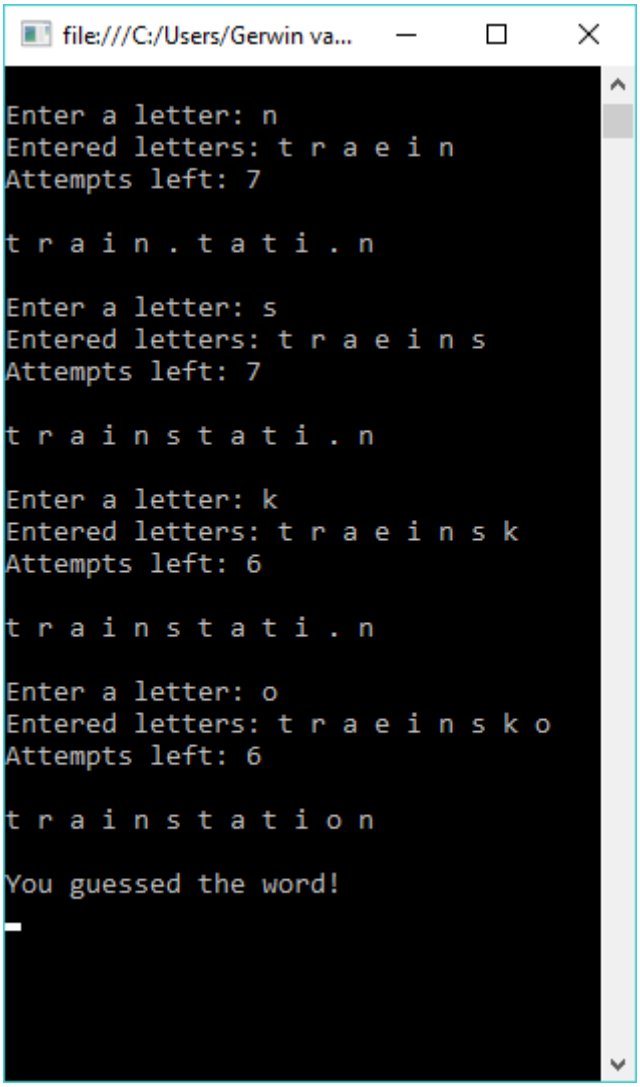
This method returns `true` if the guessed word is the same as the secret word, `false` otherwise.

j) Now we finally going to finish the implementation of the method `PlayHangman`.
→ Create a loop in method `PlayHangman` in which:

- the user can enter a new letter
- this letter is added to the list of entered letters
- the entered letters are shown
- the new letter is checked (if it is present in the hangman word)
- the attempts left is shown
- the guessed word is shown

Of course you must use the methods you created in the previous sections.
→ Make sure the loop stops when the word is guessed or the number of remaining attempts has becomes 0.
→ The `PlayHangman` method returns if the word is guessed (`true` or `false`). The `Start` method displays an appropiate message. If the user did not guess the word, then display the word.





Save your code, later we will extend this assignment
by reading words from a file.