TP C#6 : Input output.

1 Submission guidelines

You must submit an archive that complies with the following architecture :

```
– rendu-tpcs6-login_x.zip
      |– rendu-tpcs6-login_x/
            |– AUTHORS
            |– README
            |– IO
                  |– IO.sln
                        |– All except bin/ and obj/
```

You obviously need to replace *login_x* by your real login :

- The format of the AUTHORS file hasn't changed : * *login_x$* where '$' represents the line feed.
- The README file should contain the points you are not comfortable with, and it must list all the bonuses that you have done.
- Please remove the /bin and /obj folders from your submission
- An archive have been prepared for you. It is architecture compliant and contains *FIXME* code that you must complete.
- **Your code must COMPILE !**

2 Introduction

This workshop will cover file manipulation and operations on bits to do steganography.

3 Lesson

3.1 Open and close a file

In order to manipulate a file, streams are usually used. It is highly recommended to read the MSDN page about System.IO.FileStream to understand how they are working.

```
1 //new System.IO.FileStream() creates a new stream to the specified path.
2 System.IO.FileStream fs
3         = new System.IO.FileStream(path,
4         System.IO.FileMode.X,
5         System.IO.FileAccess.Y);
6         ...
7 //Do not forget to close your file stream to free the ressource
8 fs.Close();
```

You need to replace X and Y with the according properties.

3.2 Reading and writing

It is difficult to use raw streams in order to read or write in a file. Therefore, we suggest you to use the following methods in order to read or write into a stream. You must have instantiated a FileStream beforehand.

```
1 System.IO.FileStream fs_in = ...;
2 System.IO.BinaryReader br = new BinaryReader(fs_in);
3 //br.PeekChar() returns -1 when the end of the file is reached.
4 while (br.PeekChar() != -1)
5     char c = br.ReadChar(); //Reads one character
6 System.IO.FileStream fs_out = ...;
7 System.IO.BinaryWriter bw = new BinaryWriter(fs_out);
8 bw.Write("My pretty string");
```

3.3 Bitmap usage

Bitmap is a uncompressed picture file format. Even though it is not used often on the Internet, it is simple to handle and does not use lossy compression, unlike the JPEG format for instance.

Here is a small C# example of how to handle a bitmap :

```
1 // We load the picture.
2 Bitmap bmp = new Bitmap("Chemin_vers_mon_bitmap.bmp");
3 // We change the color of the pixel at (0; 0) to black.
4 bmp.SetPixel(0, 0, Color.Black);
5 // We save our modifications to another file.
6 bmp.Save("chemin_de_sortie.bmp");
```

4 Exercice 1 : Binary files

Files do not always contain text. A binary file (for instance : BMP, EXE, MP3. ...) may contain non printable characters. You cannot open them with a regular text editor such like Notepad.

However it may be interesting to extract the printable part and to isolate it from the rest of the file. The goal of this exercise is to implement a small part of the command GNU strings that does so.

> For each file given, GNU strings prints the printable character sequences that are at least 4 characters long and are followed by an unprintable character.

4.1 isPrintable

Fix the function isPrintable located in the file strings.cs, it returns True if and only if the character is printable (i.e. its ASCII code belongs to [20, 127]).

```
1 static bool isPrintable(char c);
```

4.2 strings

Fix the function strings located in the file strings.cs, it returns the result string. Every substring of the result must be followed by a line feed

```
1 static string strings(string path);
```

5 Exercise 2 : File splitter

The goal of this exercise is to implement a file splitter. It must achieve the following tasks.

— Split a file into a group of files respecting the following format : part_x.ACDC.

— The inverse operation, merge the part_.ACDC files to obtain one single file

The splitted files are denoted from 1 to n.

5.1 Split the files

The i-th character of the input file will be appended to the [i % number of files]-th file. An example may help help you to understance, an input file containing "azertiyu" will result in the creation of 3 new files which are the following :

— part_1.ACDC, containing "aru"

— part_2.ACDC, containing "zti"

— part_3.ACDC, containing "ey"

Fix the function split in the file splitter.cs, input_path is the path to the input file, dir_path is the path to the folder which will contain all the par_x.ACDC file and nb is the number of resulting files.

```
1 static void split(string input_path, string dir_path, int nb);
```

5.2 Merge the files

Implement the opposite operation, please fix the function merge in the splitter.cs source file.

```
1 static void merge(string output_path, string dir_path, int nb);
```

6 Exercise 3 : Steganography

6.1 Steganography for dummies

When one want to send a message to someone while being sure it will not be understood by someone else, it is common to use cryptography. Even though it is a well known method (HTTPS uses it in order to have a safe communication channel, for instance), it does not work well when no one must notice the communication :

When a ciphered is intercepted, it is obvious that there is something important inside it. In those case, steganography is used. It consists on trying to hide the message inside another one so that it goes unnoticed instead of making it ununderstandable (Even though the two can be combined)

6.2 Steganography in image manipulation

The goal of this exercise is to manipulate a picture in order to store a message inside it that we will be able to retrieve. To do so, we will exploit the fact that humans cannot differentiate two colors that are nearly the same. The method will therefore be the following :
Each character will be transformed into a sequence of 8 bits, and 00000000 will be added at the end.
Each bit will be hidden inside the least significant bit of the picture's pixels' components.
So, if we want to encode the binary sequence 01101000, The LSB will be set to 0 for the red component of the first pixel and the two others to one etc.
For simplicity sake, the blue component's LSB will be set to 0 and the next character wille be encoded starting from the red component of the fourth pixel.

6.3 The goal

Your mission, should you decide to accept it, is to create a program that will be able to encode and decode a message inside a picture. In order to do so, you will have to use the Bitmap class of the standard C# library (system.drawing.bitmap).
You must implement the given encoding method, but you are free to add other ones.

Get a bit from a byte

The first step for this exercise is a function that, given a byte and an index, will return the bit at this index, 0 being the MSB.

```
1 static int get_bit(byte b, int index);
```

Example : 4 ===> 00000100, get_bit(4, 5) == 1
Example : 4 ===> 00000100, get_bit(4, 7) == 0

To the next pixel

Once the three component of a pixel have been modified, we need to move to the next pixel of th picture, accounting that y must be incremented and x set back to 0 if the end of a line is reached. You should therefore code the following function :

```
1 static void set_next_coords(ref int x, ref int y, Bitmap bitmap);
```

Encoding et decoding

Your goal for this part will be to complete the code of the functions hide and reveal. They are provided in the given source archive. More detailled information about what they should do is available in the source code. The hide function must load a picture from the path provided in parameter, hide the message inside it and store the result inside the original name followed by "_out.bmp". The reveal function will load an image, extract the hidden message and return it.
Their prototypes are :

```
1 static void hide(string path, string text);
2 static string reveal(string path);
```

6.4 Bonus

In order to make the program work better, you can add anything you want, be it another graphical interface that would be better or another encoding.decoding method (That should still be available.)

Brace yourselves, Deadline is coming.