جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

CS 207 Programming methodology and abstractions

Fall 2018: midterm

Instructions:

1. The total points of this assignment are 100. The points of each exercise appear on the corresponding title.
2. Submission deadline is 15/10/2018 at 2:30 pm.
3. Use the best of your skills in code structure and naming conventions.
4. Review and consider the programming and grading guidelines presented in the syllabus including commenting and formatting.
5. Submit only source code (no executable, no screen shots). Each program file should be named after the exercise it solves. Wrap all your files in one folder and rename it as follows: yournameMidterm.zip, then zip it with the same name and submit it to the blackboard website of the course.

Exercise 1:

A string might include white spaces used to separate "tokens". These tokens can be words of a sentence, numbers or mixed data. White spaces can be spaces, tabs, newline characters, etc. As an example, the string "The boy ate the apple" contains the following tokens: "The", "boy", "ate", "the", "apple" where each is a word from an English sentence. Another example, "Chevrolet Tahoe 2016 Red 5.2L 395 XDT" contains the tokens "Chevrolet", "Tahoe", "2016", "Red", "5.2L", "395 XDT".

In this exercise, you are asked to implement the function

```
void intoTokens(string str, string* &tokens, int &nbrTokens)
```

which returns a (dynamically allocated) array of strings containing the different tokens of the string. Note that the number of tokens is initially unknown by the function. So, one needs to count the tokens before allocating the array with an appropriate size to store them.

In the file `tokens.cpp`, you are provided a program skeleton including a main function to illustrate the use of the above function. You can include any other library you may want to use.

Exercise 2:

The ppm image format is simple and human readable (if opened with a text editor). A PPM file consists of two parts, a header and the image data. The header consists of at three parts normally delimited by new line characters but the PPM specification only requires white space. The first "line" is a magic PPM identifier, it can be "P3" or "P6" (not including the double quotes!). The next line consists of the width and height of the image in pixels. The last line of the header gives the maximum value of the color intensity of the Red, Blue, and Green components for the pixels.

The image data consists in three numbers per pixel, respectively corresponding to the Red, Blue and Green light intensity of the pixel. The pixel data is written in a row wise fashion starting with the top left corner of the image. All numbers are separated by white spaces (sometimes new line characters).

Attached are two sample ppm files. Using a text editor to open each file, you will see that `feep.ppm` is a dummy image of 4 by 4 pixels (width and height on the second line of the header). The file is easily readable as text in its entirety so to give an idea of the structure of a whole ppm file. The image data consists of 48 numbers which is equivalent to 3 (R,G,B) x 4 (width) x 4 (height) and all the numbers are smaller or equal to 15 (the max value given in the third line of the header).

`blackbuck.ppm` is an actual image of 512 by 512 pixels and that's why the image data consists in many more numbers but it has exactly the same structure described above. It is possible to view the image by dragging and dropping the file in the following web viewer http://paulcuth.me.uk/netpbm-viewer/

You will also find attached a code skeleton where you should complete the missing implementation of the function `readPPM`. It takes as parameters a file path stored as a string and a variable/object of type `ppm`. It reads the image header and data from the named file and stores them in the `ppm` variable.

The code includes a main function to test the two functions. It calls the `readPPM` function to read-in the image `blackbuck.ppm`. Then, it performs a vertical flip of the image. Then, writes back the resulting image to a new file. The new file obtained could be viewed using http://paulcuth.me.uk/netpbm-viewer/ and should show an image where the animal looks to the opposite direction.