Department of Computer Science

COS284 Practical and Assignment 5: Floating Point Arithmethic and Syscalls

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

1 Introduction

This document contains both Practical 5 and Assignment 5. In general the Assignments will build upon the work of the current Practical.

1.1 Submission

The Practical will be FitchFork only and is due at 22:00 on Friday 29 October. The Assignment will also be FitchFork only and is due at 22:00 on Friday 05 November. You may use the practical sessions to ask for assistance if it is required.

1.2 Plagiarism policy

It is in your own interest that you, at all times, act responsible and ethically. As with any work done for the purpose of your university degree, remember that the University of Pretoria will not tolerate plagiarism. Do not copy a friend's work or allow a friend to copy yours. Doing so constitutes plagiarism, and apart from not gaining the experience intended, you may face disciplinary action as a result.

For more on the University of Pretoria's plagiarism policy, you may visit the following webpage: `http://www.library.up.ac.za/plagiarism/index.htm`

1.3 Practical component [25%]

1.3.1 Geometric Sequence

In this task you must implement a function, isGeometric, in assembly which determines if a sequence is geometric `https://en.wikipedia.org/wiki/Geometric_series`. The function will take two arguments, a pointer to an array of **double** and the size of the array as an **int**.

If the sequence given to the function is geometric you must return 1 else return 0. Below is some psuedo-code for the function,

```
ratio = list[1] / list[0]
for i in 0 .. size - 1:
  if list[i + 1] / list[i] != ratio:
    return 0
return 1
```

When you are finished, create an archive containing your source code file named **isGeometric.asm** and upload it to ff.cs.up.ac.za, under the **Practical 5** submission slot.

1.4 Assignment component [75%]

The assignment component deals with applying an image filter. It is recommended that you create and use smaller functions in your assembly code.

1.4.1 Background

You will be using an image format known as portable pixmap format (PPM). Details about the format can be found at `https://en.wikipedia.org/wiki/Netpbm_format`. Note that the "magic number" for the files you will be given is "P6". A brief descript of the exact image format you will be given is provided below,

```
P6
*** ***
255
RGBRGBRGBRGB...RGBRGBRGBRGB
```

Where "*** ***" represents the width and height of the image. "RGB...RGB" represents the pixel values of the image. Where, each "R", "G" and "B" represents a single unsigned byte in the image. A sequence of "RGB" represents a single pixel in the image. Note the total number of pixels in the image is equal to the width multiplied by the height of the image.

Some assumptions you may make:

- Assume the width and height will always be 3-digit numbers. All images you will be given will have both width and height in the range of $[100, 999]$. Note that width and height may not be the same.

- The maximum channel value will always be 255.
- There are no newlines in pixel data for "P6" images.
- Kernel sizes will always be a square of dimension n where $n = 2m - 1$ for $m \geq 1$.

Some additional tips:

- You can view PPM files online at `http://paulcuth.me.uk/netpbm-viewer/` if you don't have an image viewer that supports them.
- You are also given an optional script "image_converter.py" which you can use to convert images to PPM. The script also supports displaying the image. (Some OSes may not support image viewing in the script).
- You can use the command-line tool "xxd" to view the raw binary/hex values of a file.
- Each pixel is composed of 3 **unsigned char**'s. When writing these pixel channel values to the output image file ensure that you are writing **unsigned char**'s to the output file.

1.4.2 Greyscale Filter

For this task, you must convert a coloured image into a greyscale image. You will be given a function which takes two arguments. The first argument is the name of the image file you will be converting. The second is the name of the file you will save your image as. If the second file does not exist, you must create a new file (this will always be the case).

The exact way to perform the conversion is simply to take the average of the 3 channel values of a pixel ("R", "G" and "B"). Round the average to the smallest integer and use that value as the new value for all pixel channels. Below we provide some pseudo-code for the algorithm

```
char output_image[original_image.size]
copy header of original_image into output_image
for each pixel in original_image:
        r, g, b = pixel channels
        avg = (unsigned char)((r + g + b) / 3.0)
        newpixel = avg, avg, avg
        put newpixel in output_image
```

When you are finished, create an archive containing your source code file named **grey.asm** and upload it to ff.cs.up.ac.za, under the **Assignment 5** submission slot.

1.5 A note on the tasks:

For the tasks you will be given a C wrapper class that will handle input and output for you. You must simply implement the functions that will be called from inside the class and provide the correct return values. You may use the following for your assignment

```
extern open, close, read, write, malloc
```

2 Mark Distribution

Activity	**Mark**
Practical	25
Assignment	75
Total	**100**