University of Exeter

College of Engineering, Mathematics and Physical Sciences

ECM2433

The C Family

Continuous Assessment

Date Set:	28th January 2019
Date Due:	27th February 2019
Return Date:	13th March 2019

This CA comprises 15% of the overall module assessment.

This is an **individual** exercise and your attention is drawn to the College and University guidelines on collaboration and plagiarism, which are available from the College website.

This is the first coursework for this module and tests your skills and understanding of programming in C.

Please Turn Over

Problem Statement

The aim of this programming assignment is to write an ANSI standard C program that creates plots of mathematical functions in files in the Portable Network Graphics (PNG) format. There is a C library that takes care of the details of how to generate the correct format for PNG files, and you are provided with a skeleton program that produces an example image.

There are two levels of plots to be produced. The first is a simple line plot of a mathematical function of the form $y = \mathrm{f}(x)$. The second is a more complicated colour image showing a function of the form $z = \mathrm{f}(x, y)$, with the z values being represented by different colours. It is strongly suggested that you get the level 1 plots working before attempting the level 2 plots.

The program executable must be named `plotPNG`, and run using a command similar to this:

```
plotPNG testfile.png y=x^2
```

where `testfile.png` is the name of the PNG-formatted file to be output, and `y=x^2` specifies that the function to be plotted is $y = x^2$.

Task Details

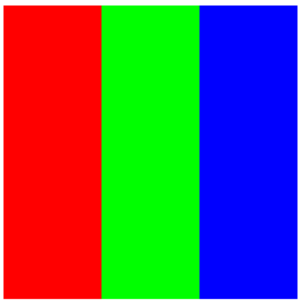
On the study resources website on ELE (`http://vle.exeter.ac.uk`), you will find a program called `plotPNG.c` which accepts a file name from the command line and outputs to that file a PNG-formatted image consisting of three vertical bars of colour. The program makes use of the `libpng` library; see `http://www.libpng.org/pub/png/libpng.html` for details, including a manual. To compile and link this program, you need to add the option "-lpng" to the link step. The following commands will perform the compile and link as two steps:

```
gcc -ansi -c plotPNG.c -I./ -o plotPNG.o
gcc plotPNG.o -lm -lpng -o plotPNG
```

You may then execute the program like this:

```
plotPNG test.png
```

This will create a new file called `test.png` containing the following image:



This particular image is made up of 300×300 pixels. The colour of each pixel is defined by an RGB (Red, Green, Blue) triplet of integers in the range 0 to 255. If all three values for a pixel are zero, then the pixel is black; if all are 255 the the pixel is white.

Your task is to change this program to output the plot of a mathematical function defined at runtime. The key function to be changed is called `makeImageData`, but you may make any changes you wish.

Figure 1 shows four plots that your program must be able to produce, with the mathematical functions given in table 1. In all of the figures the x and y axes range from 0 to 1. In figures 1c and 1d, the colour represents the z value, with the smallest z value being represented by the colour red ({255,0,0} in the RGB representation), the largest value is represented by the colour blue ({0,0,255} in the RGB representation), and values of z between these two values are assigned colours linearly between the two.

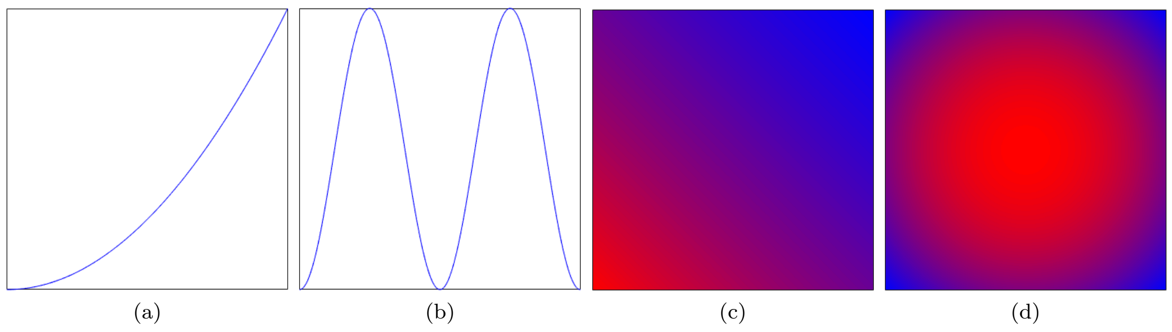


Figure 1: Four plots to be output by your program. The plots in (a) and (b) are of the nature $y = \mathrm{f}(x)$, resulting in lines. The plots in (c) and (d) are of the nature $z = \mathrm{f}(x, y)$, with the colour indicating the z value. The mathematical functions for these plots are shown in the table below.

figure	mathematical function	calculation in C and program parameter value
(a)	$y = x^2$	`y=x^2`
(b)	$y = sin^2(6.282x)$	`y=sin(6.282*x)^2`
(c)	$z = log_e(x + y + 2)$	`z=ln(x+y+2)`
(d)	$z = ((x - 0.5)^2 + (y - 0.5)^2)/2$	`z=((x-0.5)^2 + (y-0.5)^2)/2`

Table 1: The mathematical functions associated with the plots shown in figure 1.

For example, if the smallest value of z is 1 and the largest is 5, then the value 2 is $p = (2 - 1)/(5 - 1)$ of the full range and hence is represented by the following RGB values:

$$\text{red} = 255 * (1 - p) \tag{1}$$
$$\text{green} = 0 \tag{2}$$
$$\text{blue} = 255 * p \tag{3}$$

You may hard-code the four functions in table 1 into your program, and throw an error if any other function is requested. However, a better option, and one that will give you the potential for a high grade, is to look at the other program that is on the ELE page. The zipped file called `mathsExpression.zip` contains four files, as follows:

- `tinyexpr.c` and `tinyexpr.h`: a C module, written by Lewis Van Winkle (`https://github.com/codeplea`), that provides functions for parsing and evaluating mathematical functions at runtime.
- `mytest.c`: a small example C program, written by me, that uses Van Winkle's `tinyexpr` to evaluate a mathematical function passed into the program at runtime for a particular x value, or pair of x and y values, and display the calculated value.
- `comp`: a Linux shell script containing the commands needed to compile the two C modules into an executable called `mytest`.

First copy these files into your Linux directory. You will then need to make `comp` executable, by issuing the command

```
chmod u+x comp
```

and you can then compile the test program by running this script. Once compiled, you may run the program like this:

```
mathsFunction x^2
```

to calculate $y = x^2$ for $x = 3$, or:

```
mathsFunction y^2+x^2
```

to calculate $z = x^2 + y^2$ for $x = 3$ and $y = 4$. You can use the same method to enable your program to plot any function of the form $y = \mathrm{f}(x)$ or $z = \mathrm{f}(x, y)$ (within reason).

Program output

In addition to the PNG file, the program must produce two streams of output. If the program ends successfully then the message "File xxxxxx.png successfully created." should be sent to **stdout**, where "xxxxxx.png" if the name of the PNG file. Any error or warning messages must be sent to the **stderr** stream. If the program encounters a fatal error, then a meaningful message must also be sent to the **stdout** stream.

Deliverables

The deliverables for this coursework comprise both electronic and paper submissions.

Electronic submission of your C program (source code) must be made via the usual (Harrison) electronic submission process for work item "ECM2433 CA1 plotting maths functions". The program must be in ANSI standard C, and must compile, link and run on one of the Linux machines (either `emps-ugcs1` or `emps-ugcs2`). You must also submit a shell script that compiles and links your program. The executable must be called **plotPNG**.

You must provide a printed submission to the Education Office using BART, which includes the following:

- A printout of the source code for your C program. Do not include printouts of Lewis Van Winkle's `tinyexpr.c` and `tinyexpr.h` modules unless you decide to change them (there is no need to change them).
- Instructions for compiling and linking your source code into the executable.
- A short report (maximum 3 sides of A4) containing:
  - a description the design of your program, and any assumptions you have made,
  - a list summarising the errors you are trapping in your program, and
  - the output produced by your program for each of the examples shown in figure 1, even if not successfully implemented.

If your program is not completely successful in producing the required outputs, then the 3-page report is the place to describe what is not working, why you think it is not working and any testing you have done to try and locate the source of the error.

Submission must be made by 12pm (noon) on the date indicated on the front page of this document.

Suggested Approach

If this seems like a hard problem to tackle, I suggest you develop your program in the following steps, making sure each element works to your satisfaction before continuing with the next stage:

1. Compile, link and execute the version of `plotPNG.c` that I have supplied. If this does not work as described above, then contact me immediately, saying exactly what you did and the exact nature of the error.
2. Change the program to process the command line parameters (i.e. the name of the PNG file and the string containing the mathematical function) and output the values to **stdout**.
3. Create the plots shown in figures 1a and 1b.
4. Create the plots shown in figures 1c and 1d.
5. Use the `tinyexpr` functionality to enable your program to create any reasonable plot. Note that the axis limits as defined above are 0 to 1 for both x and y.

Marking Scheme

The marking scheme only applies to changes you have made to the supplied example program; code that I have supplied and you have not changed, and code created by Lewis Van Winkle, will not be marked.

Criteria	**Marks**
Program basics *Does your program compile and link without errors? Does it conform to ANSI C standards? Does it trap appropriate run-time errors and take appropriate actions? To what extent does its structure conform to the standards of a well-structured C program? To what extent does your program (1) correctly read in the command line parameters and assign the values to variables, (2) produce the specified outputs to the* ***stdout*** *and* ***stderr*** *streams, and (3) produce a meaningful PNG-formatted file containing an image that is not the example image?* Creating the example function plots shown in figures 1a and 1b *To what extent does your program successfully create the plots shown for these two functions?*	50
Creating the example function plots shown in figures 1c and 1d *To what extent does your program successfully create the plots shown for these two functions?*	20
Creating other function plots, defined at runtime *To what extent does your program successfully create appropriate plots for other functions (for example making use of `tinyexpr`)?*	20
Report *Have you included clear instructions for compiling and linking your program? To what degree does the report include and clearly describe the various elements described in the Deliverables section above? Are the spelling, grammar, language and presentation of the report clear, formal and professional and appropriate to the intended audience?*	10
Total	100

Penalties

You will be penalised 20 marks if you fail to make an electronic submission of your program code and 10 marks if you have not included evidence of the messages and output produced when your program is run for the four mandatory plots.