CM10227/50258 Principles of Programming Lab
Sheet 2

1 Introduction

Answer each question in a different file named `Lab2exNumber.c` (e.g.`Lab2ex1.c`) (i.e. one file only per exercise). We must be able to compile your code on linux.bath by typing `gcc -o Lab2ex1 Lab2ex1.c -lm`, etc. (see section below for help on logging in to `linux.bath.ac.uk` and compiling and running `C` code)

Your source (`.c`) files should be submitted in a single zip, with all files in the root of the zip: the file name should be of the form `username-labnumber.zip`, e.g. `abc12-lab2.zip`.

If you have questions about this lab sheet, or are stuck with one of the questions, talk to the tutors in the lab or post your question on Slack or in the General Q&A forum on Moodle.

This lab is part of your coursework.

Please check Moodle for the submission deadline.

2 Marking Criteria

The coursework will be conducted individually. Attention is drawn to the University rules on plagiarism. While software reuse (with referencing the source) is permitted, we will only be able to assess your contribution.

Your code will, in part, be marked by code review. In this code review the key issues for assessing the quality of the the code will be: compiling, running with expected input, robustness (handling incorrect input), code design, and the algorithms being used. You stand to lose marks for repeated or badly structured or documented code. Marks will also be deducted for ill-named variables and functions, as well as for inconsistent indentation (see examples below for a guide).

2.1 Detailed Marking Criteria

Each question is marked out of 100%. The break down of the marks are as followed,

- 10% compiling code
- 30% running with expected input
- 10% robustness (handling incorrect input)
- 30% code design and algorithm used e.g. correct use of functions, return statements, well names functions and variable
- 10% commenting

The above marking allocation accounts for 90% of the marks. The final 10% is for completing two (2) code reviews. After you have submitted this Lab Sheet you will allocated 2 other students submissions and given a code review script. Submitting these code reviews, and verification that they are a fair reflection of the quality of the code being reviewed, will mean you are awarded the 10% of marks allocated for code reviews. If you do not submit, or your code review is incomplete or judged to be of low quality by the lecturer, then you will not be allocated these marks and your submission will be capped at 50%.

2.2 Example

Example Exercise: Write a function which returns true if a number is divisible by 3 and false if it is not. Hint: use the modulo (%) operator.

2.2.1 Low Marks

```
#include <stdio.h>

int main(void) {
  int variable = 5;
  if(variable%3 == 0){
    printf("%d is divisible by 3\n", variable);
  }
  else{
    printf("%d is not divisible by 3\n", variable);
  }
  return 0;
}
```

Note: all done in the main function, poor variable names and no commenting, does not actually answer the question

2.2.2 Average Marks

```
#include <stdio.h>

// function to calculate if a number, n, is divisible by 3
int isDivisibleBy3(int n){
  if(n%3 == 0){
    return 1;
  }
  else{
    return 0;
  }
}

int main(void) {
  int n = 5;
  // calls isDivisibleBy3 function and prints out information in a nice way
  if(isDivisibleBy3(n)){
    printf("%d is divisible by 3\n", n);
  }
  else{
    printf("%d is not divisible by 3\n", n);
  }
  return 0;
}
```

Note: suitable function defined, good variable names and good commenting

2.2.3 High Marks

```
#include <stdio.h>
#include <stdbool.h>

// function to calculate if a number, n, is divisible by another number
// used stdbool.h for boolean types
bool isDivisibleBy(int n, int by){
  if(n%by == 0){
    return true;
  }
  else{
    return false;
  }
}

int main(void) {
  int n = 5;
  int by = 3;
  // if is divisible by a number print "n is divisible
  // by the variable by"
  if(isDivisibleBy(n,by)){
    printf("%d is divisible by %d\n", n, by);
  }
  else{
    printf("%d is not divisible by %d\n", n, by);
  }
  return 0;
}
```

Note: suitable functions defined, good variable names and good commenting

3 Lab Sheet Exercises

Exercise 1: Write a function(s) that prints out the reverse of a string (do not use a C standard library function to handle the reversal). Call this function from main() to test your program.

Exercise 2: Write a function(s) that converts an octal number, represented as a string (char array), into its decimal value (do not use a C standard library function to handle the conversion). Call this function from main() to test your program.

Exercise 3: Write a function(s) that prints out a tree shape (see below). The function should take two arguments: a tree width and a trunk height. For example, the arguments 9 and 4 will print out a tree of width 9 and trunk length 4, as shown below:

```
    *
   ***
  *****
 *******
*********
   ***
   ***
   ***
   ***
```

You can assume that the width of the tree will be odd and hence every line will have an odd number of asterisks. The **trunk** will always have a width of three asterisks. Call this function from main() to test your program.

Exercise 4: Write a function(s) which given a string (char array) returns true if the string is a palindrome and false if it is not. A palindrome is a word, phrase, or sequence that reads the same backwards as forwards (e.g. "rotor"; "101"). Call this function from main() to test your program.

Exercise 5: Write a function(s) to implement the *Sieve of Eratosthenes* algorithm `https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes`. This algorithm is used to find all the prime numbers less than or equal to a given integer `n`.

You should write 2 functions. One function should implement the *Sieve of Eratosthenes* algorithm using an array stored as a global variable. The second function should print whether the given integer is prime or not prime by using the generated array from the first function - this function should only accept integers between 0 and 100 (inclusive).

Finally, you should demonstrate your program by calling your functions from main().

4 Writing code & using linux.bath to compile and run code

This is a short guide to preparing, checking and submitting coursework for CM10227/50258. In order to satisfy the coursework requirements, you will need to write code, check that it compiles and runs on linux.bath and submit the original source code (.c files) to Moodle.

4.1 Using an online IDE

Perhaps the easiest way to begin programming in C, is to use an online IDE. This will simplify the process of compiling and running code as you get used to programming. One option is `https://www.codechef.com/ide`. You should download any code you wish to submit, and later test it on linux.bath prior to submission.

4.2 Using Windows and Notepad++

If you wish to code offline (our recommended approach once you are familiar with each of the steps in this document), you must use a (plain) text editor: Microsoft Word is **not** suitable for this task. One option is to use Notepad++. You can download a standalone version here: `https://notepad-plus-plus.org/downloads/`. Make sure you save your source file with a ".c" extension; once Notepad++ recognises that you're writing C code, it will start highlighting the syntax accordingly.

4.3 Using Linux (linux.bath) to write and edit code

You can also use a Linux text editor at the University to write your code on linux.bath. One example is gedit, though others exist e.g. emacs, vim, nano, ed. Once you have logged in to linux.bath (see the next section), you can use gedit by typing "gedit &" at the command prompt.

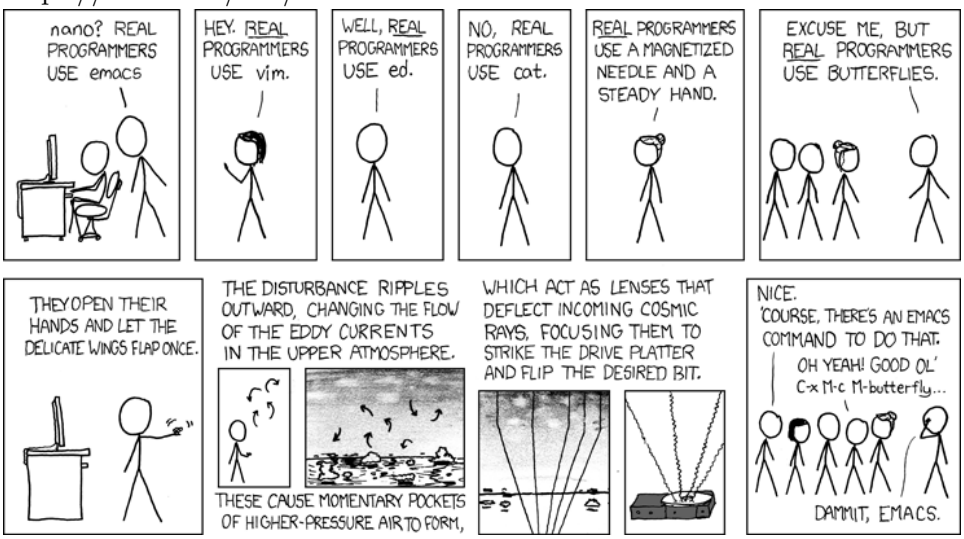
4.4 Using linux.bath to compile and run code

linux.bath is set of three servers maintained by the University, running Ubuntu 14.04. You can access these using Secure SHell (SSH). This provides a command-line interface (BaSH) and the ability to access GUI applications.

From a BUCS computer, or Unidesk, you can access linux.bath by clicking "start", typing in linux.bath and selecting the shortcut. This will start an SSH session (the first time you access, accept the certificate). If you wish to access linux.bath from your own machine, please refer to the online instructions.

The current working directory will be your "home" directory, which is conveniently the same as the "H:" drive on the BUCS computer. *View "H:"*

Figure 1: The choice of text editor is a contentious topic: https://xkcd.com/378/



on the BUCS computer (i.e. in "Computer" on the desktop), then type in "ls" to see that these are indeed the same. To get back here at any time, type in "cd ~".

Once you have written your code (using one of the methods above) and copied it to a location you can reach on linux.bath (the tutors will help with this), you can compile using gcc. It is safest to use gcc as follows, changing "HelloWorld" for your source file: *gcc -Wall -o HelloWorld HelloWorld.c -lm* (Without the "-o" flag, gcc will produce a file called a.out, and the "-Wall" turns on helpful warnings.)

Run the program by typing "./HelloWorld". The "./" is needed as it indicates that the program in the current directory must be run: by default this is not in the search path.

You can now begin programming.