An Introduction to HPC and Scientific Computing

CWM, Department of Engineering Science

University of Oxford

Wes Armour

**Practical 6: Advanced examples of CUDA programming (part one).**

This practical will review some of the example codes presented in the lecture “Introduction to the C programming language”. We will then use some of the material to build on these codes.

The learning outcomes of this practical are:

* To be familiar with basic C codes and be able to interpret them.
* To understand how variables are handled in a C code.
* To be familiar with operators and functions.
* To understand basic input and output.
* To have confidence to write your own basic C program.

This practical will be carried out on the local Linux lab machine that you are working on.

**Instructions for this practical**

***Part A***

1. Take the outline of the reduction code (reduction.cu) and
2. Add dynamic memory allocation
3. Take input from the user to determine the number of random numbers to work with.
4. f you have not done so clone the github repo for this CWM. To do this, at the command prompt type:

$ git clone https://github.com/wesarmour/CWM-in-HPC-and-Scientific-Computing.git

1. Next you should navigate to the examples directory in the prac3 folder:

$ cd CWM-in-HPC-and-Scientific-Computing/practicals/prac3/examples/

1. Now use the Makefile provided to compile all of the source codes, simply type make:

$ make

gcc example\_one.c -o example\_one

gcc example\_two.c -o example\_two

gcc example\_three.c -o example\_three

gcc example\_four.c -o example\_four

gcc example\_five.c -o example\_five

gcc example\_six.c -o example\_six

gcc example\_seven.c -o example\_seven

gcc example\_eight.c -o example\_eight

gcc example\_nine.c -o example\_nine

gcc example\_ten.c -o example\_ten

gcc example\_eleven.c -o example\_eleven

1. Look through these codes and ensure that you understand them. Execute each one to try them:

$ ./example\_seven

Circumference: 69.080002

***Part B***

It’s now time to write your first C code. Imagine the following:

A farmer has had an unfortunate visitation from some naughty aliens. They’ve destroyed some of his crop:



He now needs to work out how much of his crop has been lost. To do this he’s measured the radii (in meters) of all of the circles that have been left and stored these in a file called radii.txt:

$cd CWM-in-HPC-and-Scientific-Computing/practicals/prac3/code

He’s also tried to write a C code to calculate how much barley has been lost. He’s not really sure how to do this though so he’s included comments and not much else.

1. You should take the commented code (called barley.c) and complete the program to calculate the total loss of barley. To do this use example code eleven to guide you.
2. In previous years the farmer has found that the aliens leave rings in his field and not circles. Write a second code that calculates the loss of barley for patterns of rings.
3. Because our farmer only wants to maintain a single codebase, combine the two programs that you have written above into a single code. Think about how the farmer might use this single code, also think about future events where the aliens might create both circles and rings.
4. Add a function to your code from 7. that calculates the area of the rectangular field. You should use the scanf() function to get the dimensions as input from the farmer.
5. Add a function to your code that will calculate the total loss of barley as a percentage of the barley that is harvested from the field.
6. Finally add a function that will calculate the monetary loss to the farmer.

***Bonus questions***

1. Create a Makefile for your codes
2. Upload your own codes to your git repo

*Do not worry if you don’t complete all of the above. The aim of this practical is to encourage you to write your own C code and become familiar with some of the common functions.*