Tutorial 1: Data Structures and Algorithms
Dr Anton Gerdelan <gerdel@scss.tcd.ie>

This is a check-my-understanding type of exercise, designed to refresh your memory, and introduce you to some ideas you may not have seen before. You will be given some time to try to answer all of the questions, then we can go over them. You can work together and look things up online if you like.

1. State how large (in bytes) you expect the following data types to be on your computer. How can you check?

   char
   unsigned char
   int
   short int
   long int
   float
   double
   int*
   char*
   bool (*trick question*)

   Note: Sizes may differ on different machine architectures! Later versions of C/C++ have headers with types at exact sizes if you need them.

2. a) State which section of program memory (heap, stack, etc) each variable is stored in.
   b) I put some **`static`** keywords in as a talking point. `static` is a poorly-named keyword in C. What do they mean in each case?
   c) What is the relationship of variables to the block section?
   d) What do you expect the output of each `printf()` statement to be?

```
#include<stdlib.h> // malloc()
#include<stdio.h> // printf()

static int g_grand_total;

static int more(int input) {
  static int accum = 1;
  return input + accum++;
}

int main() {
  int first_val = 10;
  int second_val = 22;
  g_grand_total += first_val;
```

```
  { // this is a block
    int third_val = 1;
    g_grand_total += (second_val + third_val);
  }

  int* finals = NULL;
  // equivalent to C++ "finals = new int[3]";
  finals = (int*)malloc(sizeof(int) * 3);
  finals[0] = first_val;
  finals[1] = second_val;
  finals[2] = g_grand_total;

  printf("finals = %p\n", finals);
  printf("&finals[0] = %p\n", &finals[0]);
  printf("&finals[1] = %p\n", &finals[1]);
  printf("&finals[2] = %p\n", &finals[2]);

  free(finals);
  return 0;
}
```

3. Adapted from *R. Sedgewick, "Algorithms in C++", Addison-Wesley, 1992*:
   A classic elementary problem is reducing a fraction **u** / **v** to it's smallest terms.

   i.e. 5 / 10 has a greatest common denominator (gcd) of 5. Divide both **u** and v by **5** to get 1 / 2. How do we find the gcd?

   Euclid's Algorithm was devised by the ancient Greeks to solve this problem. As written by Euclid in *Elements*, we know that if **u** is greater than **v** then the greatest common divisor (gcd) of **u** and **v** is the same as the gcd of **v** and **u - v**.

```
int gcd( int u, int v ){
  int t;
  while( u > 0 ){
    if ( u < v ){ t = u; u = v; v = t; }
    u = u - v;
  }
  return v;
}
```

   Q. Do you follow this and know how to **step** through this on paper (on in a debugger)?
   3 a) Is it possible to reduce the number of iterations by replacing `u=u-v` with u=`u%v`? Check that it produces the same results, even with negative inputs.
   3 b) Write a function to reduce a given fraction to its lowest terms, using:

```
struct Fraction {
  int numerator;
  int denominator
};
```

3 c) Give all values that u,v take on if gcd() is initialised with gcd(12345, 56789);

3 d) How many C statements are executed for the previous function call?

3 e*) Write a function to compute the gcd of 3 inputs integers; u, v, w.

3 f**) Find the largest pair of integers possible in your system with gcd of 1.