Explore Memory

> This is a **pair** exercise and must be competed in your **tutorial** or **lab** with your partner.

This activity is to experiment with memory and see where your C compiler stores variables and how much memory it uses for different types.

In this activity, you will look at various *types* used to store data in C. We call these *types* the *primitive types*. The types, and the **%** sequence needed to display them in `printf`, are:

- `int` (**%d**)
- `unsigned int` (**%u**)
- `short` (**%hd**)
- `unsigned short` (**%hu**)
- `char` (**%hhd**)
- `unsigned char` (**%hhu**)
- `long` (**%ld**)
- `unsigned long` (**%lu**)
- `long long` (**%lld**)
- `unsigned long long` (**%llu**)
- `float` (**%f**)
- `double` (**%lf**)

Each of these types can actually be used to store numbers.

Important note

You will need to use `gcc` for this activity, *not* `dcc`.

This is because `dcc` has all sorts of neat features to stop you from accidentally making certain types of mistakes in your programs – but for this exercise, we *want* to make those "mistakes", to see what's really happening.

If you are curious, you could try compiling your program with both `dcc` **and** `gcc`, and comparing the differences.

Steps

1. Download and save **explore.c**
2. Edit the program to conduct experiments to answer the questions below

3. Record the answers on your blog – if you prefer, you could write them in a text file (e.g. using `gedit`) as you go, and then copy to your blog when you're finished.

In my solution I put in lots of printfs to display information I needed such as the addresses of variables

Q1: Size of types

Use `sizeof` find out the number of bytes that C uses on your system to store the various types used in the program below.

Q2: Location of variables

Print out the address of the various variables used in the program and see if you can deduce how C places them in memory, e.g. what things are placed near each other, what things are placed far away?

Are adjacent variables placed next to each other in memory or are there gaps? (consider the results you found using sizeof above to help you answer this)

Q3: Overflowing and Underflowing

What are the maximum and minimum values you can store in each of the types in the program below? What happens when you edit the program below to overflow them by 1, and what happens when you underflow them by 1? Cut and paste output from the edited program showing what you discover.

Blog

Write up a blog post detailing your discoveries. Write down the size, minimum, and maximum for each type and explain where in memory the variaous variables were located as well as any other interesting findings.

To run Styl-o-matic:

```
$ 1511 stylomatic explore.c
Looks good!
```

You'll get advice if you need to make changes to your code.

Submit your work with the *give* command, like so:

```
$ give cs1511 wk04_exploreMemory
```

Or, if you are working from home, upload the relevant file(s) to the `wk04_exploreMemory` activity on Give Online.