Note: this lab is basically asking you to redo the Feb. 10 lecture, but with blocks of integers instead of blocks of characters. It is possible to just copy the code from the lecture and modify it slightly, but it makes more sense to write the code from scratch.

Question 1.

In this question, you will implement a structure known as `IntArrayList`, a structure similar to Python's `list` (with the difference that it can only store C integers).

The idea is to expand the memory block used to store the data if you are running out of space. You can use `realloc` to get a new pointer to an area of memory which has the requested size, and to which the old contents would be copied if necessary.

`ArrayList` would keep track of the size of the current list (i.e., the number of elements in it) as well as the capacity (i.e., the maximum possible size).

When appending, you should expand capacity by a factor of 2 using `realloc` if you are running out of space.

Part (a)

Read the documentation for `memmove` (note: need `string.h` for `memmove`) and `memcpy`. Why is `memcpy` an inappropriate choice when implementing `list_insert` and `list_delete`?

Part (b)

Implement and test the `ArrayList` functions, as declared in `intlist.h`.

Part (c)

Create a `main.c` file that would test the `ArrayList` functions.
We will not check off untested code. Testing and debugging is part of the work. It is always better to debug functions one at a time rather doing it at the end.

You must compile your code using VS Code's tasks.json file.

Part (d)

Demonstrate to a TA that you can trace your code in VS Code by putting in a breakpoint in `main.c` and stepping through the code.

Question 2.

Now, you will create a different C file, `intlist2.c`. In `intlist2.c`, change the way the capacity is managed: only increase the capacity by 1 int at a time, and shrink the capacity by a factor of 2 whenever that is possible.

Make sure you can compile `main.c` together with `intlist2.c`, and, separately, with `intlist.c`. Explain why doing both at the same time is impossible.

Question 3.

Rewrite `intlist.c` and `intlist2.c` so that there is as little repeated code as possible. You should accomplish that by writing a third C file which contains common code, and creating helper functions that can be called from the old functions in `intlist.c` and `intlist2.c`.