Tips & tricks - Week 1

C Compiler

We will be using GCC (GNU C Compiler) which is available for most systems.

- Windows
  - The Cygwin project
  - The MinGW and mingw-w64 projects.
- MacOS
  - Use the Homebrew package manager
- Linux
  - If GCC isn't already installed, it is available through the distro's package manager
  - Debian/Ubuntu: `apt install gcc`
  - Arch: `pacman -S gcc`
  - Redhat/Fedora: `yum install gcc`

Compiling tips

Compile & Run

To compile and run a C program, use the following command to compile the file to an executable file

```
gcc -o <output_file> <input_file>
```

and to run the file, use

```
./<output_file>
```

Makefiles

- Makefiles are used to automate the build process
- They define *targets* which can be specific files to be built or actions to be performed
- We will use Makefiles to ease the amount of commands we need to type to compile and run our programs
- Makefiles must be called **Makefile** or **makefile** and are written in plain text A simple Makefile could look like this:

```
target1:
    command1
    command2

target2:
    command1
    command3
```

You can then *make* the desired target by running `make target1` which will then execute command1 and then command2

Exercise tips

Good to know libraries:

- `limits.h`: Defines minimum and maximum values for different types. Such as `INT_MIN`, `INT_MAX` & `UINT_MAX`
- `float.h`: Defines minimum and maximum values for different floating-point types and precisions. Such as `DBL_MIN` and `DBL_MAX`
- `sys/param.h`: Has some *nice to use* macros such as `MIN(a,b)` and `MAX(a,b)` as well as bitmask operations (might be useful in later exercises)
- `math.h`: Implements a lot of math functions such as `sqrt(x)`, `pow(x,y)`, `sin(x)` and `cos(x)`. Is not linked by default, so you must add the linker flag `-lm` to `gcc` when compiling code using `math.h`

C Cheat Sheet

Allocations

- Requires the `stdlib.h` header
- `malloc(size_t size)`: Allocates a block of memory of size `size` and returns a pointer to the first byte of the block
- `calloc(size_t num, size_t size)`: Allocates a block of memory for an array of `num` elements of `size` bytes each and returns a pointer to the first byte of the block. Also sets the allocated memory region to 0s.

I/O operations

- Requires the `stdio.h` header
- `printf(const char *format, ...)`: Prints the formatted string to stdout
  - `%d`: Signed decimal integer
  - `%u`: Unsigned decimal integer
  - `%f`: Decimal floating point
  - `%c`: Character
  - `%s`: String
- `fscanf(FILE *stream, const char *format, ...)`: Reads formatted input from `stream` and stores them according to the given format.
  - Open a file with `fopen(const char *filename, const char *mode)`
  - Close a file with `fclose(FILE *stream)`
  - `%d`: Signed decimal integer
  - `%u`: Unsigned decimal integer
  - `%f`: Decimal floating point
  - `%c`: Character
  - `%s`: String
  - `%[^\n]`: String until newline
  - `%[^\t]`: String until tab
  - `%5[0-9]`: String of length 5 containing only digits