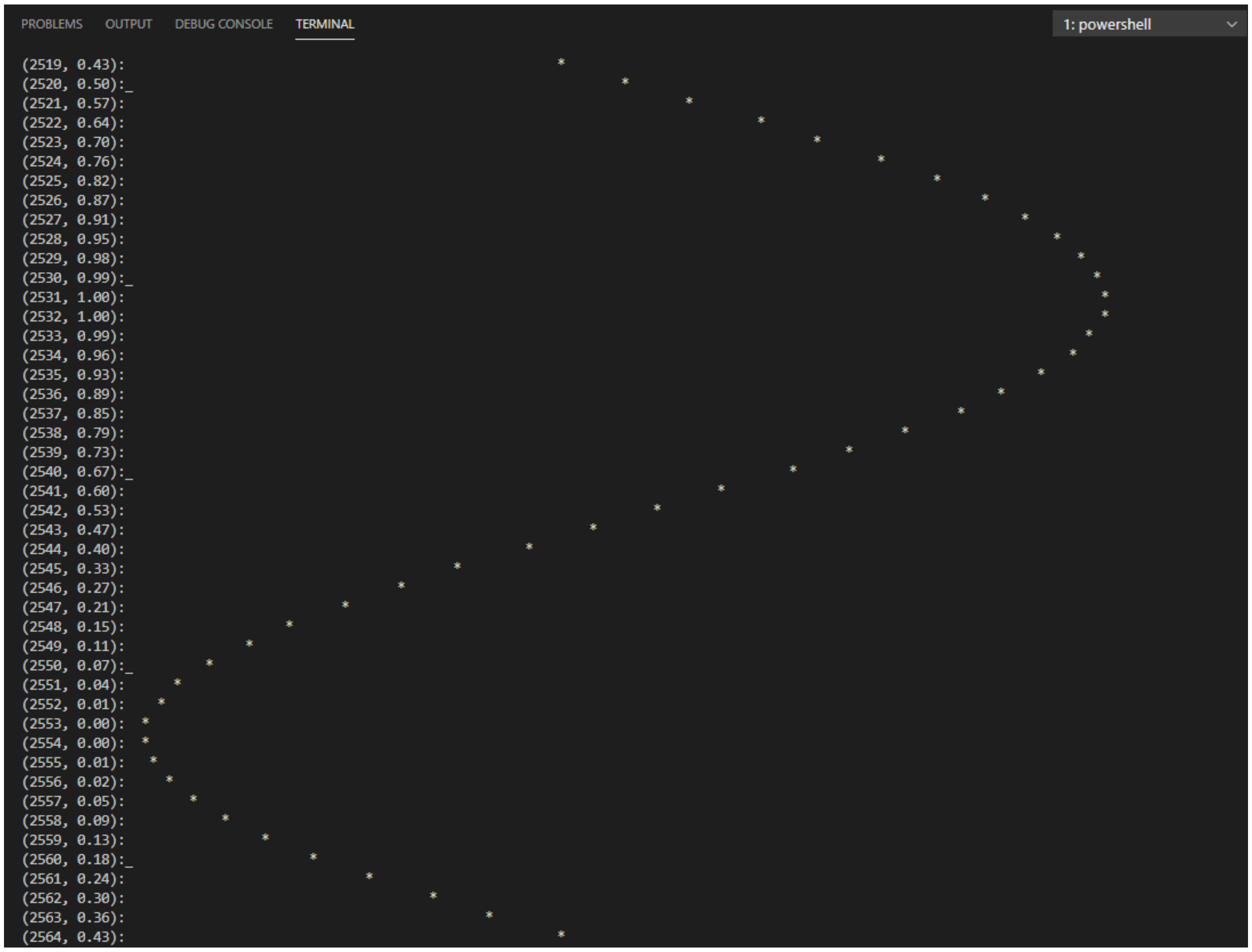
Hosted C Programming Logbook
Joseph Butterworth



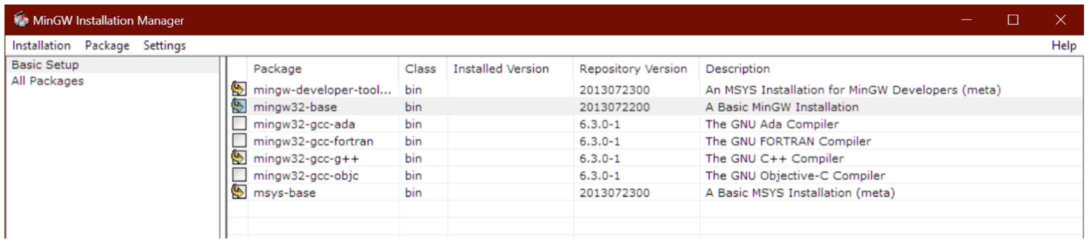
```
(2519, 0.43):                                                        *
(2520, 0.50):_                                                               *
(2521, 0.57):                                                                        *
(2522, 0.64):                                                                                 *
(2523, 0.70):                                                                                        *
(2524, 0.76):                                                                                                *
(2525, 0.82):                                                                                                       *
(2526, 0.87):                                                                                                             *
(2527, 0.91):                                                                                                                   *
(2528, 0.95):                                                                                                                       *
(2529, 0.98):                                                                                                                          *
(2530, 0.99):_                                                                                                                          *
(2531, 1.00):                                                                                                                             *
(2532, 1.00):                                                                                                                             *
(2533, 0.99):                                                                                                                           *
(2534, 0.96):                                                                                                                         *
(2535, 0.93):                                                                                                                     *
(2536, 0.89):                                                                                                                *
(2537, 0.85):                                                                                                          *
(2538, 0.79):                                                                                                   *
(2539, 0.73):                                                                                            *
(2540, 0.67):_                                                                                    *
(2541, 0.60):                                                                            *
(2542, 0.53):                                                                    *
(2543, 0.47):                                                            *
(2544, 0.40):                                                    *
(2545, 0.33):                                          *
(2546, 0.27):                                   *
(2547, 0.21):                            *
(2548, 0.15):                     *
(2549, 0.11):                *
(2550, 0.07):_          *
(2551, 0.04):       *
(2552, 0.01):     *
(2553, 0.00):   *
(2554, 0.00):   *
(2555, 0.01):    *
(2556, 0.02):      *
(2557, 0.05):         *
(2558, 0.09):             *
(2559, 0.13):                  *
(2560, 0.18):_                       *
(2561, 0.24):                               *
(2562, 0.30):                                       *
(2563, 0.36):                                              *
(2564, 0.43):                                                        *
```

Contents

Setting Up the Hosted Toolchain Software

GCC (the compiler) requires an emulation layer on windows, called MinGW. To start the installation download the MinGW installer and run it. Select continue for Step 1 and Step 2.

When the Installation Manager appears, select the following packages:



	Package	Class	Installed Version	Repository Version	Description
✓	mingw-developer-tool...	bin		2013072300	An MSYS Installation for MinGW Developers (meta)
✓	mingw32-base	bin		2013072200	A Basic MinGW Installation
☐	mingw32-gcc-ada	bin		6.3.0-1	The GNU Ada Compiler
☐	mingw32-gcc-fortran	bin		6.3.0-1	The GNU FORTRAN Compiler
✓	mingw32-gcc-g++	bin		6.3.0-1	The GNU C++ Compiler
☐	mingw32-gcc-objc	bin		6.3.0-1	The GNU Objective-C Compiler
✓	msys-base	bin		2013072300	A Basic MSYS Installation (meta)

Select *Apply Changes* from the *Installation* menu.

After this add the /bin folders of MinGw and MSYS to the Environment Path Variable:

- Right click on start button and select *Search*
- Type **`system env`**
- Select *Environment Variables*
- In the System variables find the *Path* Variable
- Click edit and append the install locations of MinGW and MSYS.
- With a standard install these would be: **`C:\MinGW\bin`** and **`C:\MinGW\msys\1.0\bin`**
- Click ok and then again to confirm completeThis is done so that other programs know where to find MinGW executables.

Verify your install by:

- Open the start menu and type "cmd" and select command prompt.
- Then type gcc --help and press enter
- You see a table with some help text

Using gcc:

- You can open a command prompt in a folder by:
  - going to the folder in "File Explorer"
  - clicking in the address bar
  - deleting all text
  - typing "cmd" and pressing enter.
- Create a c file with your editor in the folder
- Go back to the command prompt and run `gcc YOUR_FILE_NAME.c -o YOUR_FILE_NAME`
- To run your program run `YOUR_FILE_NAME`

C1: Introduction to C Programming - Toolchain
https://secure.ecs.soton.ac.uk/notes/ellabs/1/c1/c1.pdf
https://en.wikibooks.org/wiki/C_Programming/Variables
https://gcc.gnu.org/onlinedocs/gcc-3.0.1/cpp_4.html

Preparation - 06/10/20

	Listing 1	Listing 2	Listing 3
No error message	✓		
An error from the preprocessor		✓	
An error from the compiler			✓
An error from the linker			

Can declare multiple variables in one line as long as they are the same type.

```
int anumber, anothernumber, yetanothernumber;
```

Can assign it a value

```
int some_new_number = 4;
```

Can make multiple variables have the same value

```
anumber = anothernumber = yetanothernumber = 8;
```

A variable can be named anything as long as it is made of letters, numbers and underscores. It must begin with a letter.

int is an integer that can be stored as a signed 32 bit number.

char is a character in the ASCII character set. When defining a char it can be expressed as the character or its corresponding ASCII number.

float is an inexact representation of a real number, meaning it isn't always 100% accurate. It can store decimals, and numbers much greater and much smaller than an int can.

double is a double precision float, meaning it is a float that can store twice as much (floats on steroids!). It takes up more space than a float so is not preferable when space is at a minimum.

sizeof function will tell you how much memory is actually used by any variable.

```
size_t size;
int i;
size = sizeof(i);
```

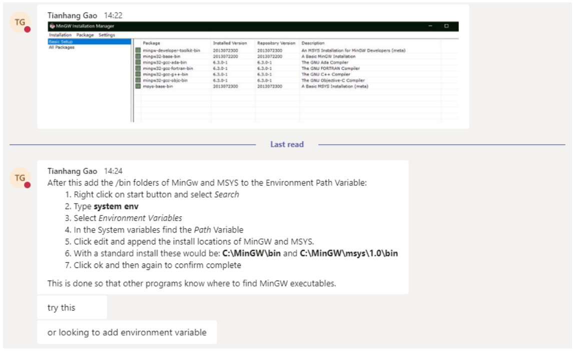
const qualifier is used before a data type, for example const int. This is used at the start of your code to define a variable, it is then not allowed to change.

const int may be used to define a magic number, this is a variable that will stay the same within the code but may be needed to change between versions.

#define may also be used to define a magic number. #define is used within the preprocessor so if used incorrectly can cause more damage to the code than const int.

Hello World - 07/10/20

Cannot use command line, suspect c compiler installed incorrectly. Reinstalled minGW with all repositories. Then add a path to the environment variable.



Tianhang Gao 14:22

Package	Installed Version	Repository Version	Description
mingw-developer-toolkit-bin	2013072300	2013072300	An MSYS Installation for MinGW Developers (meta)
mingw32-base-bin	2013072200	2013072200	A Basic MinGW Installation
mingw32-gcc-ada-bin	6.3.0-1	6.3.0-1	The GNU Ada Compiler
mingw32-gcc-fortran-bin	6.3.0-1	6.3.0-1	The GNU FORTRAN Compiler
mingw32-gcc-g++-bin	6.3.0-1	6.3.0-1	The GNU C++ Compiler
mingw32-gcc-objc-bin	6.3.0-1	6.3.0-1	The GNU Objective-C Compiler
msys-base-bin	2013072300	2013072300	A Basic MSYS Installation (meta)

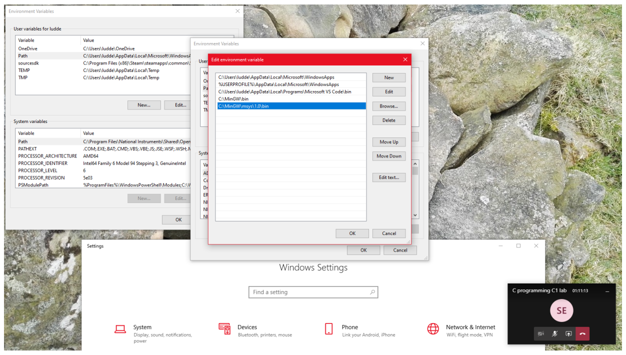
Tianhang Gao 14:24
After this add the /bin folders of MinGw and MSYS to the Environment Path Variable:

1. Right click on start button and select *Search*
2. Type **system env**
3. Select *Environment Variables*
4. In the System variables find the *Path* Variable
5. Click edit and append the install locations of MinGW and MSYS.
6. With a standard install these would be: **C:\MinGW\bin** and **C:\MinGW\msys\1.0\bin**
7. Click ok and then again to confirm complete

This is done so that other programs know where to find MinGW executables.

try this

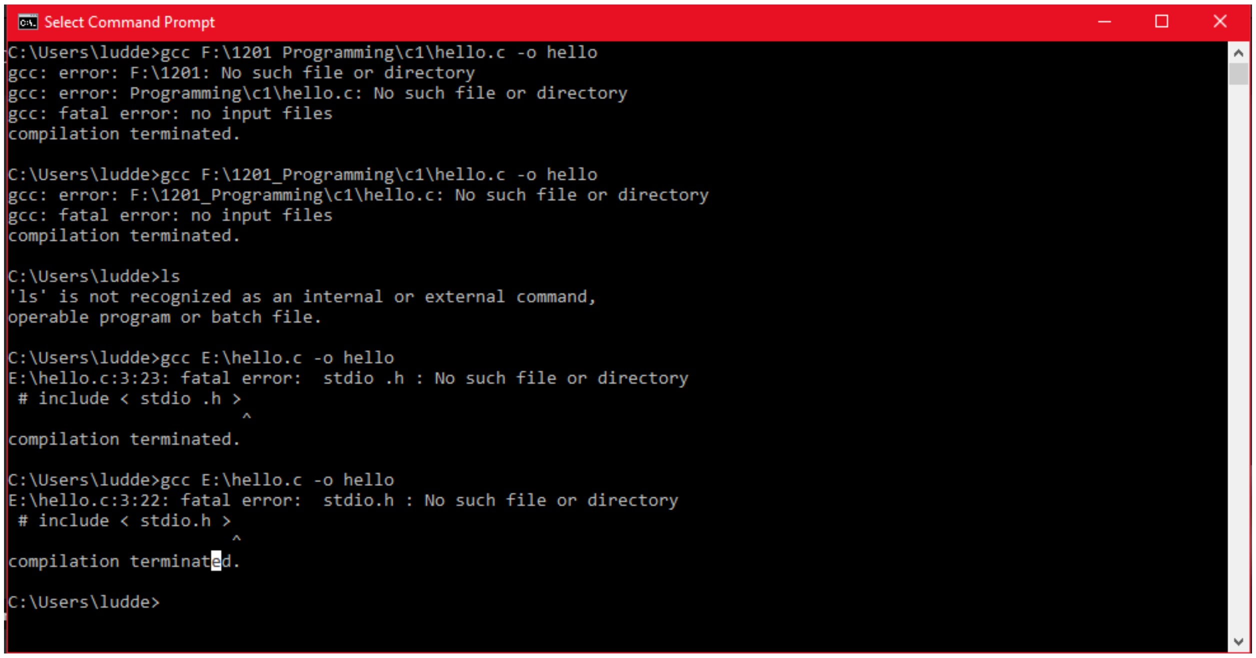
or looking to add environment variable



Edit environment variable

```
C:\Users\ludde\AppData\Local\Microsoft\WindowsApps
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\Users\ludde\AppData\Local\Programs\Microsoft VS Code\bin
C:\MinGW\bin
C:\MinGW\msys\1.0\bin
```

Initially unable to find the path to hello.c so moved it to E:\ making it easy to access. The code when copied did not compile as spaces where added to the stdio.h directory.

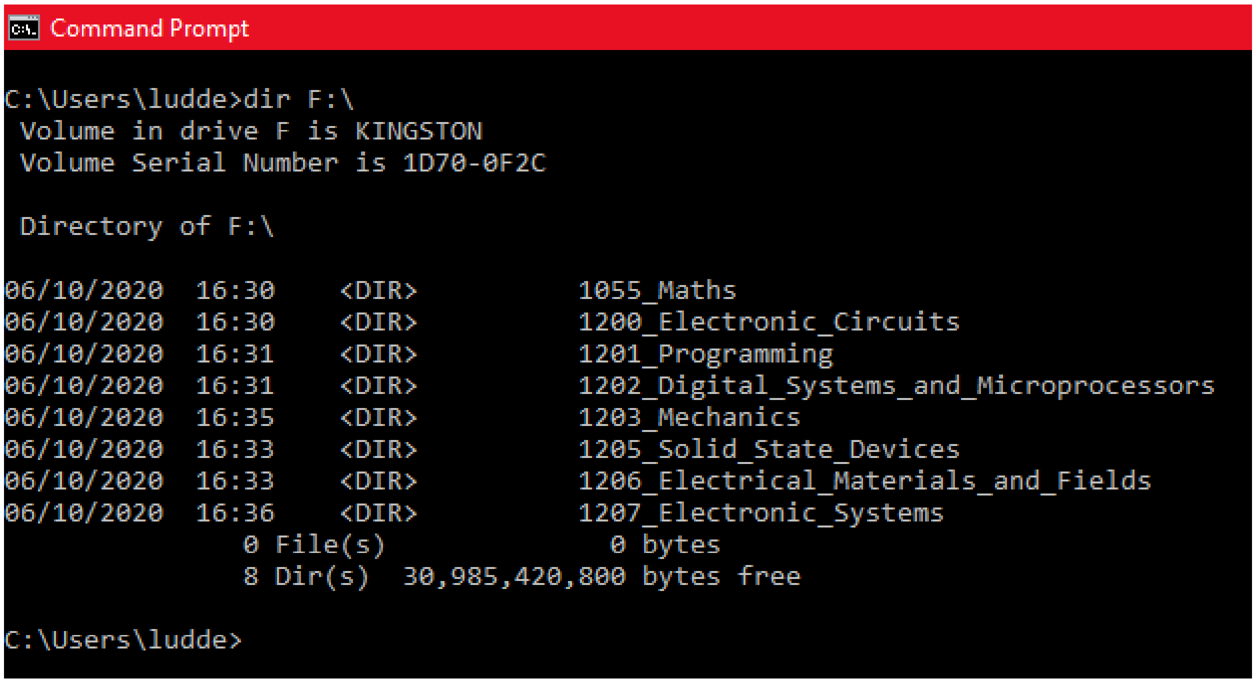


Select Command Prompt

```
C:\Users\ludde>gcc F:\1201 Programming\c1\hello.c -o hello
gcc: error: F:\1201: No such file or directory
gcc: error: Programming\c1\hello.c: No such file or directory
gcc: fatal error: no input files
compilation terminated.

C:\Users\ludde>gcc F:\1201_Programming\c1\hello.c -o hello
gcc: error: F:\1201_Programming\c1\hello.c: No such file or directory
gcc: fatal error: no input files
compilation terminated.

C:\Users\ludde>ls
'ls' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\ludde>gcc E:\hello.c -o hello
E:\hello.c:3:23: fatal error:  stdio .h : No such file or directory
 # include < stdio .h >
                       ^
compilation terminated.

C:\Users\ludde>gcc E:\hello.c -o hello
E:\hello.c:3:22: fatal error:  stdio.h : No such file or directory
 # include < stdio.h >
                      ^
compilation terminated.

C:\Users\ludde>
```

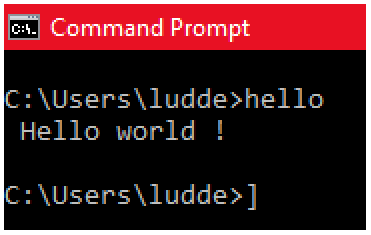
Cannot allow spaces in file or folder names as this will be read as a break by the terminal, and will not path to your file properly.

Using the dir command I am able to find all the files and folders in a directory.



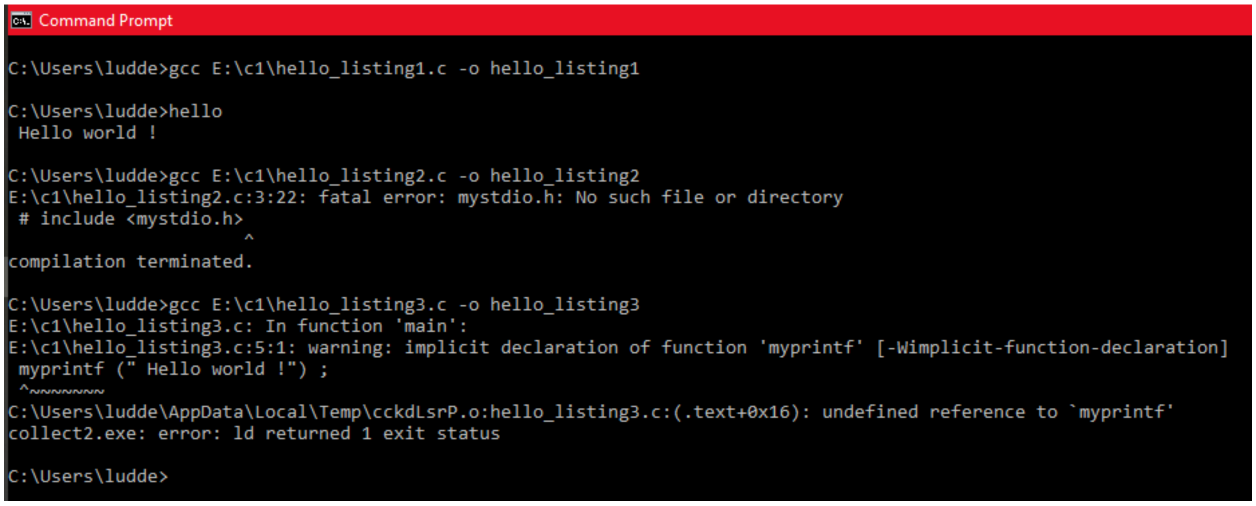
```
C:\Users\ludde>dir F:\
 Volume in drive F is KINGSTON
 Volume Serial Number is 1D70-0F2C

 Directory of F:\

06/10/2020  16:30    <DIR>          1055_Maths
06/10/2020  16:30    <DIR>          1200_Electronic_Circuits
06/10/2020  16:31    <DIR>          1201_Programming
06/10/2020  16:31    <DIR>          1202_Digital_Systems_and_Microprocessors
06/10/2020  16:35    <DIR>          1203_Mechanics
06/10/2020  16:33    <DIR>          1205_Solid_State_Devices
06/10/2020  16:33    <DIR>          1206_Electrical_Materials_and_Fields
06/10/2020  16:36    <DIR>          1207_Electronic_Systems
               0 File(s)              0 bytes
               8 Dir(s)  30,985,420,800 bytes free

C:\Users\ludde>
```

Now that the c has been compiled it is running on the command line. When I type hello, because of the program, the command line responds with “Hello World!”.



```
C:\Users\ludde>hello
 Hello world !

C:\Users\ludde>]
```

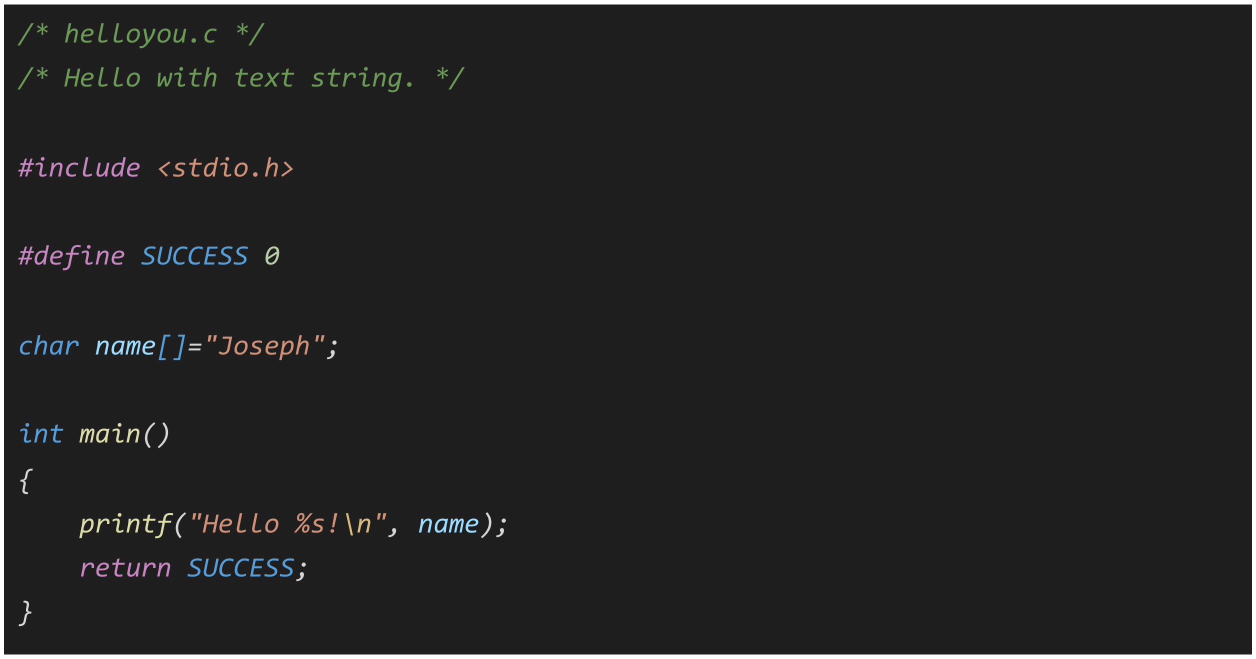
The first listing works fine and is able to be compiled. This means “Hello world!” responded.
If I were to remove the “*/” from the second line then the entire set of code would be commented and there would be nothing to compile.
The second listing had the wrong library in the code this means that when the preprocessor tried to install it it was unable to. This resulted in an error message of “No such file or directory”.
The third listing was not able to be compiled in the linker because it used a different print function. This could not be compiled because this function was not defined.


Command Prompt

```
C:\Users\ludde>gcc E:\c1\hello_listing1.c -o hello_listing1

C:\Users\ludde>hello
 Hello world !

C:\Users\ludde>gcc E:\c1\hello_listing2.c -o hello_listing2
E:\c1\hello_listing2.c:3:22: fatal error: mystdio.h: No such file or directory
 # include <mystdio.h>
                      ^
compilation terminated.

C:\Users\ludde>gcc E:\c1\hello_listing3.c -o hello_listing3
E:\c1\hello_listing3.c: In function 'main':
E:\c1\hello_listing3.c:5:1: warning: implicit declaration of function 'myprintf' [-Wimplicit-function-declaration]
 myprintf (" Hello world !") ;
 ^~~~~~~~
C:\Users\ludde\AppData\Local\Temp\cckdLsrP.o:hello_listing3.c:(.text+0x16): undefined reference to `myprintf'
collect2.exe: error: ld returned 1 exit status

C:\Users\ludde>
```

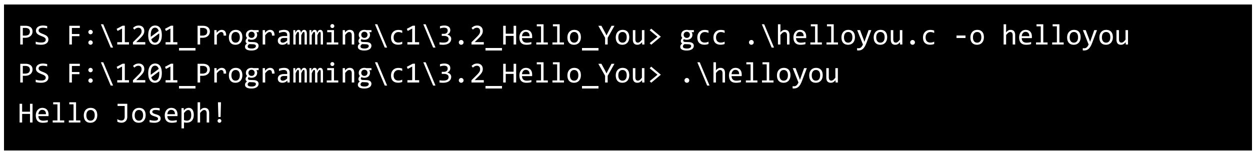
My predictions from the preparation activity were relatively accurate. I was able to predict what the errors in the code would be for listings 1 & 2. I was incorrect in saying there would be an error in the compiler.

Hello You - 8/10/20



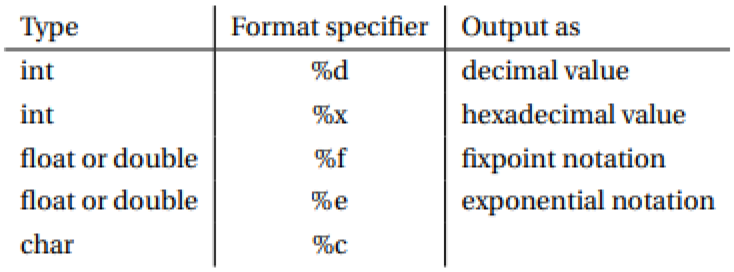
```c
/* helloyou.c */
/* Hello with text string. */

#include <stdio.h>

#define SUCCESS 0

char name[]="Joseph";

int main()
{
    printf("Hello %s!\n", name);
    return SUCCESS;
}
```

With the hello you program I was able to edit the variable so that the program would call me by my name (or whatever name I gave it).



```
PS F:\1201_Programming\c1\3.2_Hello_You> gcc .\helloyou.c -o helloyou
PS F:\1201_Programming\c1\3.2_Hello_You> .\helloyou
Hello Joseph!
```

In my text editor I tested switching the format specifier for my name's variable. I tested this for all the format types in the table opposite. I found that my name's variable worked with every format specifier. However, it would change the output values depending on the specifier.



Type	Format specifier	Output as
int	%d	decimal value
int	%x	hexadecimal value
float or double	%f	fixpoint notation
float or double	%e	exponential notation
char	%c	

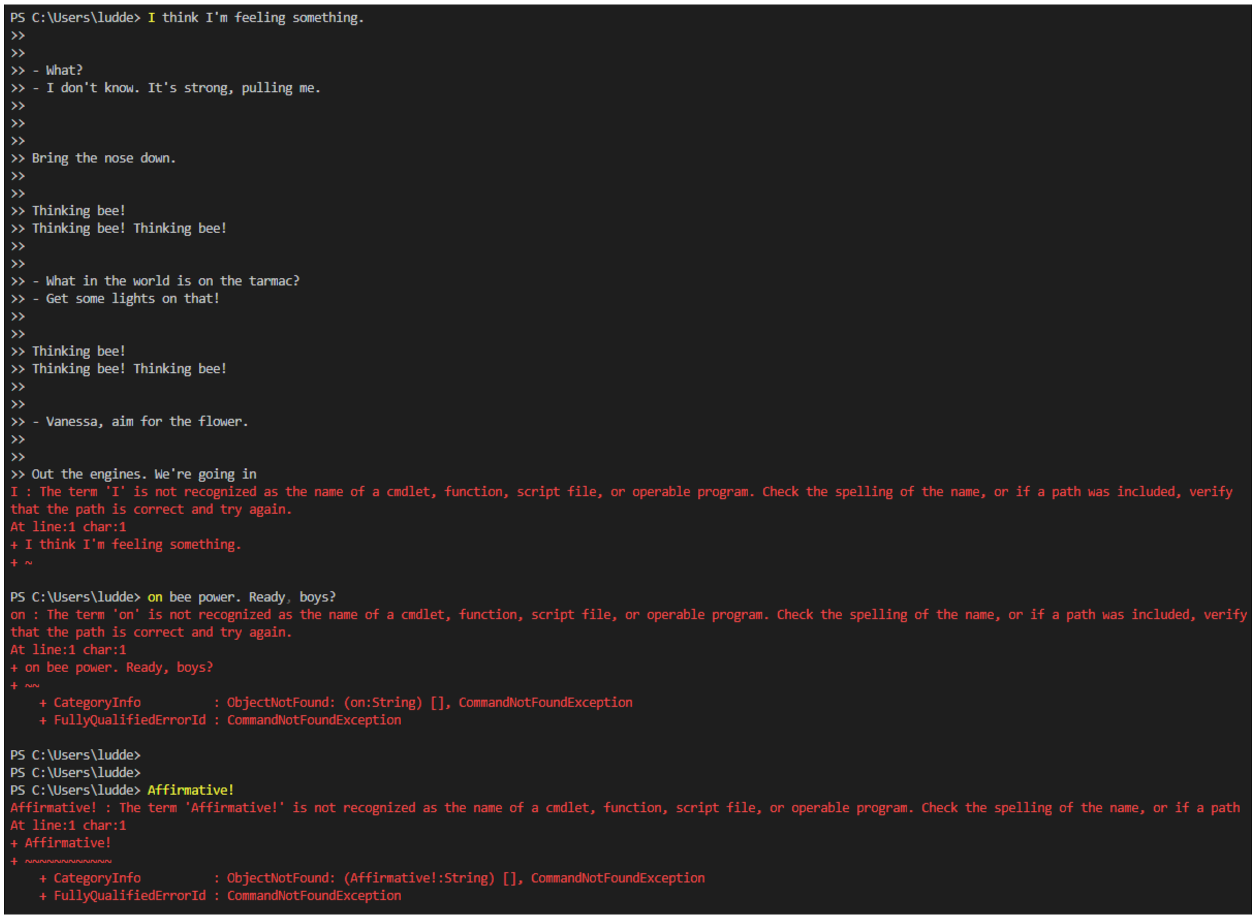
```
PS F:\1201_Programming\c1\3.2_Hello_You> gcc .\helloyou.c -o helloyou
PS F:\1201_Programming\c1\3.2_Hello_You> .\helloyou
Hello 4210692!
PS F:\1201_Programming\c1\3.2_Hello_You> gcc .\helloyou.c -o helloyou
PS F:\1201_Programming\c1\3.2_Hello_You> .\helloyou
Hello 404004!
PS F:\1201_Programming\c1\3.2_Hello_You> gcc .\helloyou.c -o helloyou
PS F:\1201_Programming\c1\3.2_Hello_You> .\helloyou
Hello 0.000000!
PS F:\1201_Programming\c1\3.2_Hello_You> gcc .\helloyou.c -o helloyou
PS F:\1201_Programming\c1\3.2_Hello_You> .\helloyou
Hello 1.501925e-307!
PS F:\1201_Programming\c1\3.2_Hello_You> gcc .\helloyou.c -o helloyou
PS F:\1201_Programming\c1\3.2_Hello_You> .\helloyou
Hello ♦!
```

Hello Input - 8/10/20

```c
/* hello-input.c */
/* Ask for input. */

#include <stdio.h>

#define SUCCESS 0
#define NAME_BUFFSIZE 100

int main()
{
    char name[NAME_BUFFSIZE];

    printf("What is your name?  ");
    scanf("%s", name);
    printf("Hello %s!\n", name);
    return SUCCESS;
}
```

```
PS F:\1201_Programming\c1\3.3_Hello_Input> gcc .\hello_input.c -o
.\hello_input
PS F:\1201_Programming\c1\3.3_Hello_Input> .\hello_input
What is your name?  Joseph
Hello Joseph!
```

I was successfully able to test “hello input”. It worked, successfully taking my inputted name. In line 11 memory space is reserved for the user input. If the input is larger than the reserved space then the input will overflow the allocated memory and will write in an uncontrolled way to adjacent memory.



```
PS C:\Users\ludde> I think I'm feeling something.
>>
>>
>> - What?
>> - I don't know. It's strong, pulling me.
>>
>>
>>
>> Bring the nose down.
>>
>>
>> Thinking bee!
>> Thinking bee! Thinking bee!
>>
>>
>> - What in the world is on the tarmac?
>> - Get some lights on that!
>>
>>
>> Thinking bee!
>> Thinking bee! Thinking bee!
>>
>>
>> - Vanessa, aim for the flower.
>>
>>
>> Out the engines. We're going in
I : The term 'I' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path was included, verify
that the path is correct and try again.
At line:1 char:1
+ I think I'm feeling something.
+ ~

PS C:\Users\ludde> on bee power. Ready, boys?
on : The term 'on' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path was included, verify
that the path is correct and try again.
At line:1 char:1
+ on bee power. Ready, boys?
+ ~~
    + CategoryInfo          : ObjectNotFound: (on:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\ludde>
PS C:\Users\ludde>
PS C:\Users\ludde> Affirmative!
Affirmative! : The term 'Affirmative!' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path
At line:1 char:1
+ Affirmative!
+ ~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (Affirmative!:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException
```

With a long input the characters will overflow the program and be read through the command line.

Debug - 15/12/20

The compiler also defines constants for the programmer, one of which is __LINE__, it contains the current line number. I then defined a statement called MARK that inserts a (printf()) call that prints the current line number.

```c
/* debug.c */
/* Debugs Hello_Input.c */

#include <stdio.h>

#define SUCCESS 0
#define NAME_BUFFSIZE 100
#define MARK printf("This is line %d \n", __LINE__)

int main()
{
    char name[NAME_BUFFSIZE];

    printf("What is your name?  ");
    scanf("%s", name);
    MARK;
    printf("Hello %s!\n", name);

    return SUCCESS;
}
```

```
PS F:\1201_Programming\c1\XMAS\4_Debug> gcc .\debug.c -o .\debug
PS F:\1201_Programming\c1\XMAS\4_Debug> .\debug
What is your name?  Joseph
This is line 16
Hello Joseph!
```

I then changed the program so that it would only print the line number when DEBUG is defined as 1. This allows me to turn the line printing on and off between compiles to find errors in my code.

```c
/* debug.c */
/* Debugs Hello_Input.c */

#include <stdio.h>

#define SUCCESS 0
#define NAME_BUFFSIZE 100
#define DEBUG 1
#define MARK printf("This is line %d \n", __LINE__)

int main()
{
    char name[NAME_BUFFSIZE];

    printf("What is your name?  ");
    scanf("%s", name);

    #if DEBUG == 1
    MARK;
    #endif

    printf("Hello %s!\n", name);

    return SUCCESS;
}
```

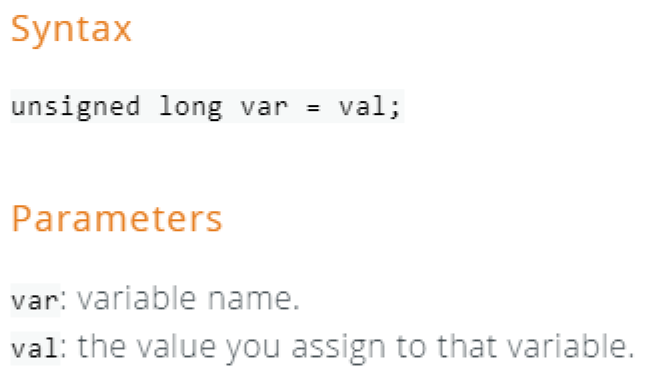
```
PS F:\1201_Programming\c1\XMAS\4_Debug> gcc .\debug.c -o .\debug
PS F:\1201_Programming\c1\XMAS\4_Debug> .\debug
What is your name?  Joseph
This is line 19
Hello Joseph!
PS F:\1201_Programming\c1\XMAS\4_Debug> gcc .\debug.c -o .\debug
PS F:\1201_Programming\c1\XMAS\4_Debug> .\debug
What is your name?  Joseph
Hello Joseph!
```

C2: Functions and Control Flow
https://secure.ecs.soton.ac.uk/notes/ellabs/1/c2/c2.pdf
https://www.arduino.cc/reference/en/language/variables/data-types/unsignedlong/
https://en.wikipedia.org/wiki/C_data_types
https://en.wikipedia.org/wiki/Printf_format_string
https://stackoverflow.com/questions/20186809/endless-loop-in-c-c
https://stackoverflow.com/questions/3585846/color-text-in-terminal-applications-in-unix/23657072#23657072
https://wiki.bash-hackers.org/scripting/terminalcodes
https://www.tutorialspoint.com/c_standard_library/c_function_rand.htm

Preparation - 13/10/20

1. Notes about declaring an unsigned long variable, how to initialize it correctly, and what format specifier is required to print such a data type.

An unsigned long variable can be declared like so:



Syntax

```
unsigned long var = val;
```

Parameters

`var`: variable name.
`val`: the value you assign to that variable.

To initialize an unsigned long variable I have to make sure the compiler knows what data type I am using. When I declare my variable I must put the type at the end of the initialized value.

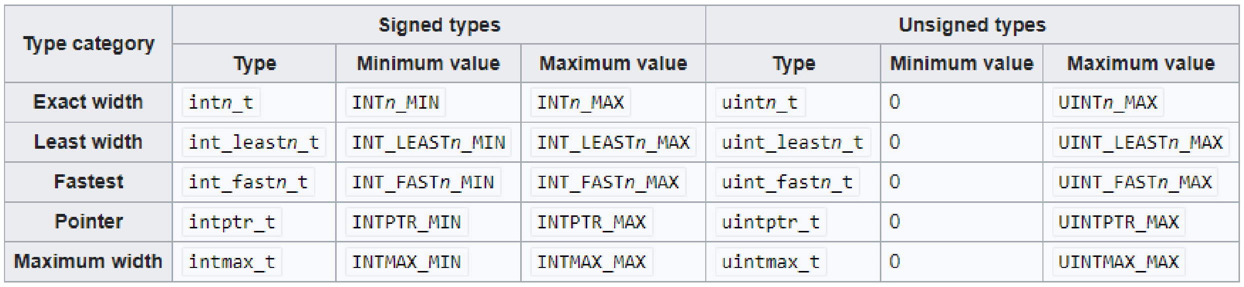
Unsigned long var = 10UL;

The format specifier for an unsigned long variable is %lu.

Fixed width data types are data types that have a specified length, because every system has its own default length This helps the portability and reliability of the code. To use a fixed width data type I must include the <stdint.h> library, and if using macros the <inttype.h> library.

2. Notes on how to print a number with a fixed character width.

When declaring the variable and wanting a fixed character width value I must use the library's data types:



Type category	Signed types			Unsigned types		
	Type	Minimum value	Maximum value	Type	Minimum value	Maximum value
Exact width	`intn_t`	`INTn_MIN`	`INTn_MAX`	`uintn_t`	0	`UINTn_MAX`
Least width	`int_leastn_t`	`INT_LEASTn_MIN`	`INT_LEASTn_MAX`	`uint_leastn_t`	0	`UINT_LEASTn_MAX`
Fastest	`int_fastn_t`	`INT_FASTn_MIN`	`INT_FASTn_MAX`	`uint_fastn_t`	0	`UINT_FASTn_MAX`
Pointer	`intptr_t`	`INTPTR_MIN`	`INTPTR_MAX`	`uintptr_t`	0	`UINTPTR_MAX`
Maximum width	`intmax_t`	`INTMAX_MIN`	`INTMAX_MAX`	`uintmax_t`	0	`UINTMAX_MAX`

n is the fixed width of the data type, for example

int_fast8_t

Is the fastest data type of length 8.

Within the printf() and scanf() format strings the fixed width format specifier is %d.

3. Notes on how to create an endless loop in C.

To create an endless loop I can use the for or while commands. I can use the for command to create an endless loop like so:

```
for (;;) {
...
}
```

> is an "infinite" loop, presumably to be broken by other means, such as a break or return. Whether to use while or for is largely a matter of personal preference.

Using the while command to create the endless loop can be done like so:

while(1){}

However this is likely to be misidentified as an error by the compiler because the statement is always true.

4. A formula that gives you the sine of a value scaled to the range of 0 to 1. (You should also have an idea how to change the frequency in your formula).

To create a sine function I can sin() function in the <math.h> library. This gives the radian sine value of an input. To increase the frequency I can give the input a coefficient.

The sine value can be scaled to the range of 0 to 1 using the formula y=(sin(x)+1)/2

5. A draft of the plotval() function needed in Part 2. Consider to use a loop to print an appropriate number of spaces into the same line.

My version of the plotval function:

```c
/*plotval.*/

/*libraries*/
#include <stdio.h>
#include <stdint.h>
#include <math.h>

/*definitions*/
#define MAX_TERMINAL 120

/*plotval function*/
int plotval(float ploti)                                    /*Declare
plotval function and plot input*/
{
    for (int i = 0; i <= roundf(ploti * MAX_TERMINAL); i++)     /*Loop till at
proportion of the screen to sin value*/
    {
        printf(" ");
    }
    printf("*\n");                                          /*Mark with an
asterix*/
}
```

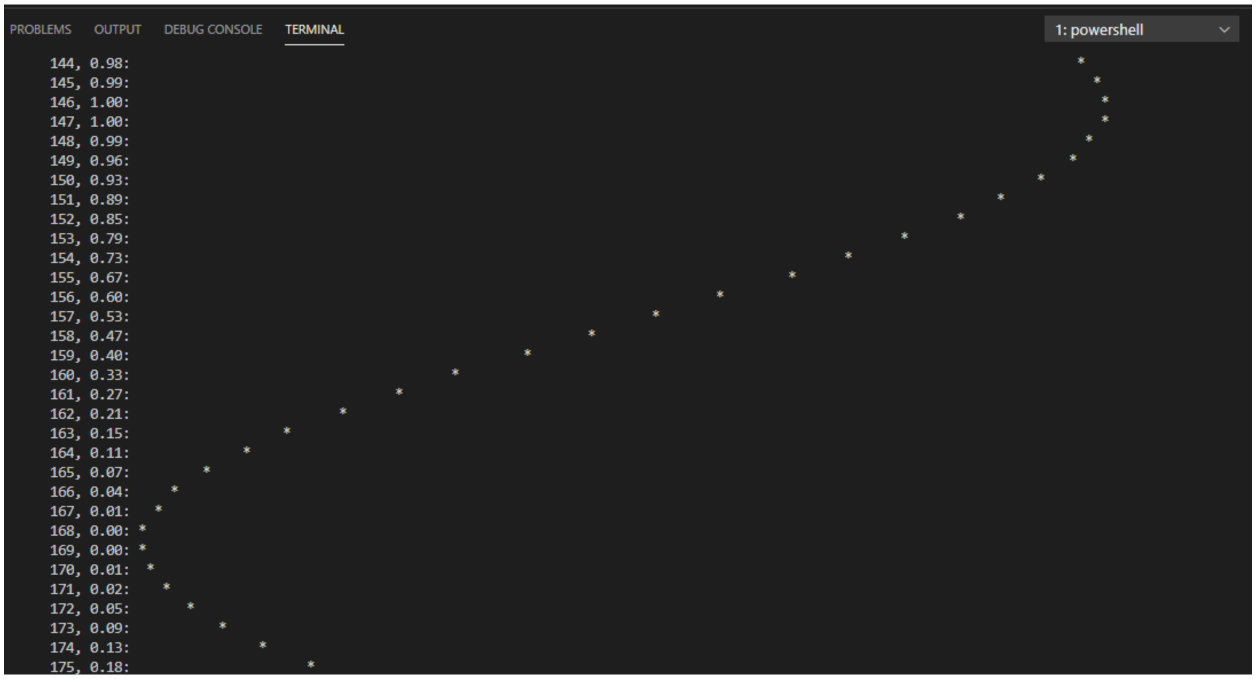
int main() - 14/10/20

I started my code by having my loop print the values of x on consecutive lines.

```
/* sine_plotter.c */

#include <stdio.h>
#include <stdint.h>
#include <math.h>

#define SUCCESS 0

int main() {
   uint32_t x = 0;           /*Declare x as fixed width unsigned variable,
initialise equal to 0*/
for(;;){                     /*Endless loop*/
    x++;                     /*Increase x by 1*/
    printf("%d\n", x);       /*Print value of x to command line*/
}
    return SUCCESS;
}
```

int plotval() - 14/10/20
I modified int main() so my program would calculate the value of y, being the sine of x. And then translate the value of y to the range 0 and 1. I then added the plotval() function that I made during my preparation for this lab. This created a trace of the sine function on the terminal.

```
/* sine_plotter.c */

#include <stdio.h>
#include <stdint.h>
#include <math.h>

#define SUCCESS 0
#define pi 3.14159265358979
#define FREQ 5
#define MAX_TERMINAL 120

int plotval(float);
int main()
{
    uint32_t x = 0; /*Declare x as fixed width unsigned variable, initialise
equal to 0*/
    float y = 0;
    for (;;) /*Endless loop*/
    {
        x++;                                    /*Increase x by 1*/
        printf("%8.0d, ", x);                   /*Print value of x to command
line*/
        y = (sin(FREQ * x * pi / 180) + 1) / 2; /*calculate value of y*/
        printf("%0.2f:", y);                    /*Print value of y to command
line*/
        plotval(y);
    }
    return SUCCESS;
}

int plotval(float ploti)
{
    for (int i = 0; i <= roundf(ploti * MAX_TERMINAL); i++)
    {
        printf(" ");
    }
    printf("*\n");
}
```

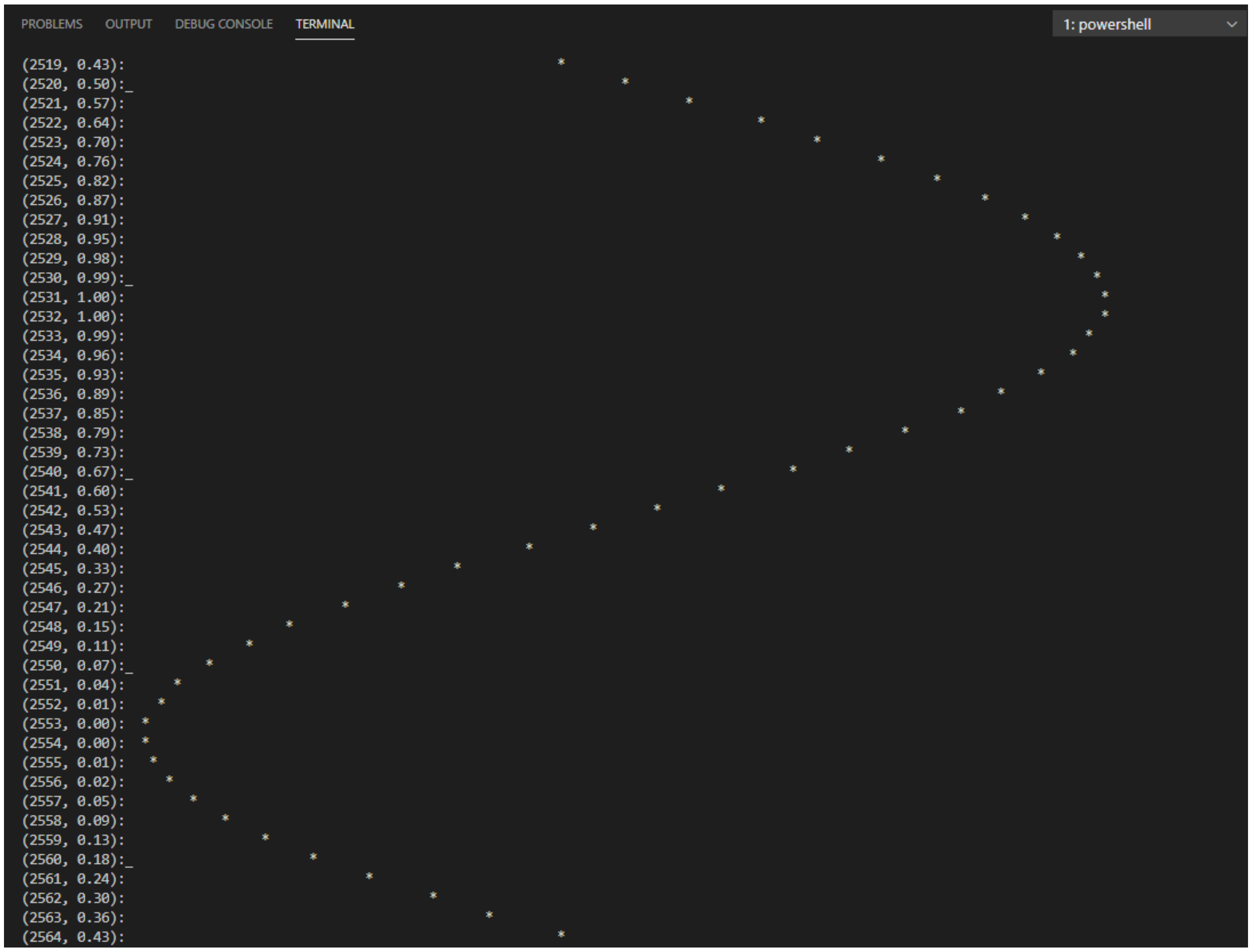


```
     144, 0.98:                                                                                                                      *
     145, 0.99:                                                                                                                        *
     146, 1.00:                                                                                                                         *
     147, 1.00:                                                                                                                         *
     148, 0.99:                                                                                                                       *
     149, 0.96:                                                                                                                     *
     150, 0.93:                                                                                                                 *
     151, 0.89:                                                                                                            *
     152, 0.85:                                                                                                       *
     153, 0.79:                                                                                                *
     154, 0.73:                                                                                         *
     155, 0.67:                                                                                  *
     156, 0.60:                                                                         *
     157, 0.53:                                                                 *
     158, 0.47:                                                         *
     159, 0.40:                                                 *
     160, 0.33:                                        *
     161, 0.27:                                 *
     162, 0.21:                          *
     163, 0.15:                   *
     164, 0.11:              *
     165, 0.07:         *
     166, 0.04:     *
     167, 0.01:   *
     168, 0.00: *
     169, 0.00: *
     170, 0.01:  *
     171, 0.02:    *
     172, 0.05:       *
     173, 0.09:           *
     174, 0.13:                *
     175, 0.18:                      *
```

Tick-Marks - 14/10/20
I then added a trace to the screen for scale. I added this every 10 lines like the provided example but this could easily be modified by changing the value in the if statement.

After this I changed the size of my fixed character width and added brackets so that it would neatly display the coordinates of the sine function.

```
/*Main code*/
int main()
{
    uint32_t x = 0;                             /*Declare x as fixed width
unsigned variable, initialise equal to 0*/
    float y = 0;
    for (;;)                                    /*Endless loop*/
    {
        x++;                                    /*Increase x by 1*/
        printf("(%8.0d, ", x);                   /*Print value of x to command
line*/
        y = (sin(FREQ * x * pi / 180) + 1) / 2; /*calculate value of y*/
        printf("%0.2f):", y);                    /*Print value of y to command
line*/
        if (x % 10 == 0)                        /*If multiple of 10 ad tick*/
        {
            printf("_");
        }
        else                                    /*Otherwise add space, so
function is aligned*/
        {
            printf(" ");
        }
        plotval(y);                             /*Call Plotval*/
    }
    return SUCCESS;
}
```

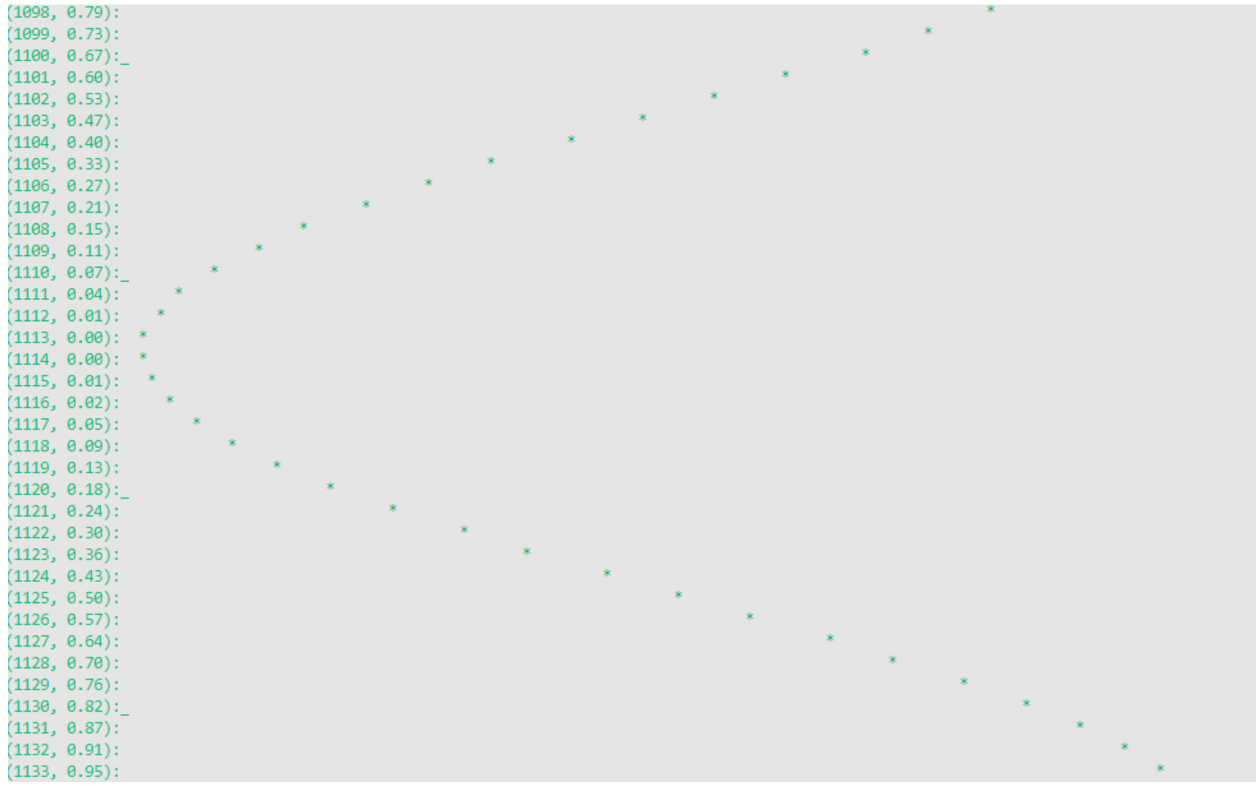


```
(2519, 0.43):                                                       *
(2520, 0.50):_                                                              *
(2521, 0.57):                                                                       *
(2522, 0.64):                                                                                *
(2523, 0.70):                                                                                       *
(2524, 0.76):                                                                                               *
(2525, 0.82):                                                                                                      *
(2526, 0.87):                                                                                                            *
(2527, 0.91):                                                                                                                 *
(2528, 0.95):                                                                                                                     *
(2529, 0.98):                                                                                                                        *
(2530, 0.99):_                                                                                                                         *
(2531, 1.00):                                                                                                                            *
(2532, 1.00):                                                                                                                            *
(2533, 0.99):                                                                                                                          *
(2534, 0.96):                                                                                                                        *
(2535, 0.93):                                                                                                                    *
(2536, 0.89):                                                                                                              *
(2537, 0.85):                                                                                                         *
(2538, 0.79):                                                                                                  *
(2539, 0.73):                                                                                           *
(2540, 0.67):_                                                                                  *
(2541, 0.60):                                                                           *
(2542, 0.53):                                                                   *
(2543, 0.47):                                                           *
(2544, 0.40):                                                   *
(2545, 0.33):                                          *
(2546, 0.27):                                   *
(2547, 0.21):                            *
(2548, 0.15):                     *
(2549, 0.11):                *
(2550, 0.07):_          *
(2551, 0.04):       *
(2552, 0.01):     *
(2553, 0.00):   *
(2554, 0.00):   *
(2555, 0.01):    *
(2556, 0.02):      *
(2557, 0.05):         *
(2558, 0.09):             *
(2559, 0.13):                  *
(2560, 0.18):_                       *
(2561, 0.24):                               *
(2562, 0.30):                                       *
(2563, 0.36):                                              *
(2564, 0.43):                                                       *
```

The speed at which my program executes will be determined on the clock speed of the system running it. Visual Studio Code, my IDE, will have a clock speed that will do so many instructions a second. On embedded systems this will vary on the device. An arduino allows you to control the clock speed that you use to run the code through.

Colours & Different Trig Functions - 14/10/20

I then implemented the ability to change the foreground and background colours. It has been commented so I know which number corresponds to which number.



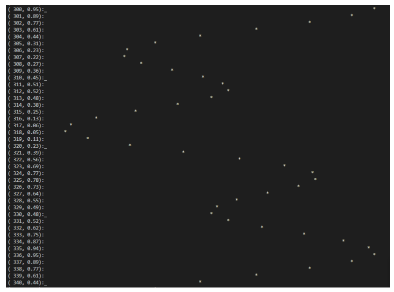
```
(1098, 0.79):                                                                                                  *
(1099, 0.73):                                                                                           *
(1100, 0.67):_                                                                                   *
(1101, 0.60):                                                                           *
(1102, 0.53):                                                                   *
(1103, 0.47):                                                           *
(1104, 0.40):                                                   *
(1105, 0.33):                                          *
(1106, 0.27):                                   *
(1107, 0.21):                            *
(1108, 0.15):                     *
(1109, 0.11):               *
(1110, 0.07):_         *
(1111, 0.04):      *
(1112, 0.01):    *
(1113, 0.00):  *
(1114, 0.00):  *
(1115, 0.01):   *
(1116, 0.02):     *
(1117, 0.05):        *
(1118, 0.09):            *
(1119, 0.13):                 *
(1120, 0.18):_                      *
(1121, 0.24):                              *
(1122, 0.30):                                      *
(1123, 0.36):                                             *
(1124, 0.43):                                                      *
(1125, 0.50):                                                              *
(1126, 0.57):                                                                      *
(1127, 0.64):                                                                               *
(1128, 0.70):                                                                                      *
(1129, 0.76):                                                                                              *
(1130, 0.82):_                                                                                                    *
(1131, 0.87):                                                                                                           *
(1132, 0.91):                                                                                                                *
(1133, 0.95):                                                                                                                    *
```

With The combination of the two sine functions with different frequencies and phases I was able to create different functions for it to plot. This required a more complex formula to be used. This can combine two sine waves together.

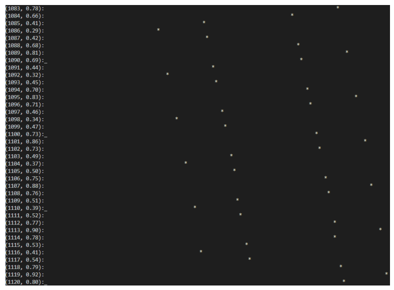
```
/* sine_plotter.c */

/*Libraries*/
#include <stdio.h>
#include <stdint.h>
#include <math.h>

/*Definitions*/
#define SUCCESS 0
#define pi 3.14159265358979
#define MAX_TERMINAL 120
#define FREQ_A 10
#define FREQ_B 30
```

```
#define PHASE_A 5
#define PHASE_B 90
#define BACK_COL "\x1B[39m" /*Colour solution from David Guyon on
StackOverflow*/
#define FORE_COL "\x1B[49m" /*0=black, 1=red, 2=green, 3=yellow, 4=blue,
5=magenta, 6=cyan, 7=white*/
int plotval(float);

/*Main code*/
int main()
{
    uint32_t x = 0; /*Declare x as fixed width unsigned variable, initialise
equal to 0*/
    float y = 0;

    for (;;) /*Endless loop*/
    {
        x++;
/*Increase x by 1*/
        printf(BACK_COL FORE_COL "(%4d, ", x);
/*Print value of x to command line*/
        y = (sin(((FREQ_A * x) + PHASE_A) * pi / 180)+sin(((FREQ_B * x) +
PHASE_B) * pi / 180)+2)/4; /*calculate value of y*/
        printf("%0.2f):", y);
/*Print value of y to command line*/
        if (x % 10 == 0)
/*If multiple of 10 ad tick*/
        {
            printf("_");
        }
        else /*Otherwise add space, so function is aligned*/
        {
            printf(" ");
        }
        plotval(y); /*Call Plotval*/
    }
    return SUCCESS;
}
```

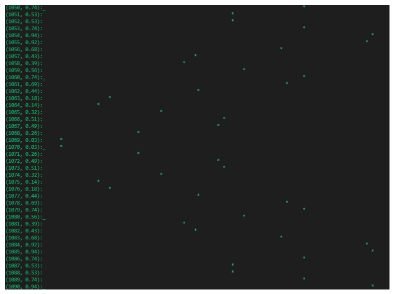
```c
/*plotval function*/
int plotval(float ploti) /*Declare plotval function and plot input*/
{
    for (int i = 0; i <= roundf(ploti * MAX_TERMINAL); i++) /*Loop till at proportion of the screen to sin value*/
    {
        printf(" ");
    }
    printf("*\n"); /*Mark with an asterix*/
}
```



```
( 300, 0.95):_
( 301, 0.89):
( 302, 0.77):
( 303, 0.61):
( 304, 0.44):
( 305, 0.31):
( 306, 0.23):
( 307, 0.22):
( 308, 0.27):
( 309, 0.36):
( 310, 0.45):_
( 311, 0.51):
( 312, 0.52):
( 313, 0.48):
( 314, 0.38):
( 315, 0.25):
( 316, 0.13):
( 317, 0.06):
( 318, 0.05):
( 319, 0.11):
( 320, 0.23):_
( 321, 0.39):
( 322, 0.56):
( 323, 0.69):
( 324, 0.77):
( 325, 0.78):
( 326, 0.73):
( 327, 0.64):
( 328, 0.55):
( 329, 0.49):
( 330, 0.48):_
( 331, 0.52):
( 332, 0.62):
( 333, 0.75):
( 334, 0.87):
( 335, 0.94):
( 336, 0.95):
( 337, 0.89):
( 338, 0.77):
( 339, 0.61):
( 340, 0.44):_
```



```
(1083, 0.78):
(1084, 0.66):
(1085, 0.41):
(1086, 0.29):
(1087, 0.42):
(1088, 0.68):
(1089, 0.81):
(1090, 0.69):_
(1091, 0.44):
(1092, 0.32):
(1093, 0.45):
(1094, 0.70):
(1095, 0.83):
(1096, 0.71):
(1097, 0.46):
(1098, 0.34):
(1099, 0.47):
(1100, 0.73):_
(1101, 0.86):
(1102, 0.73):
(1103, 0.49):
(1104, 0.37):
(1105, 0.50):
(1106, 0.75):
(1107, 0.88):
(1108, 0.76):
(1109, 0.51):
(1110, 0.39):_
(1111, 0.52):
(1112, 0.77):
(1113, 0.90):
(1114, 0.78):
(1115, 0.53):
(1116, 0.41):
(1117, 0.54):
(1118, 0.79):
(1119, 0.92):
(1120, 0.80):_
```



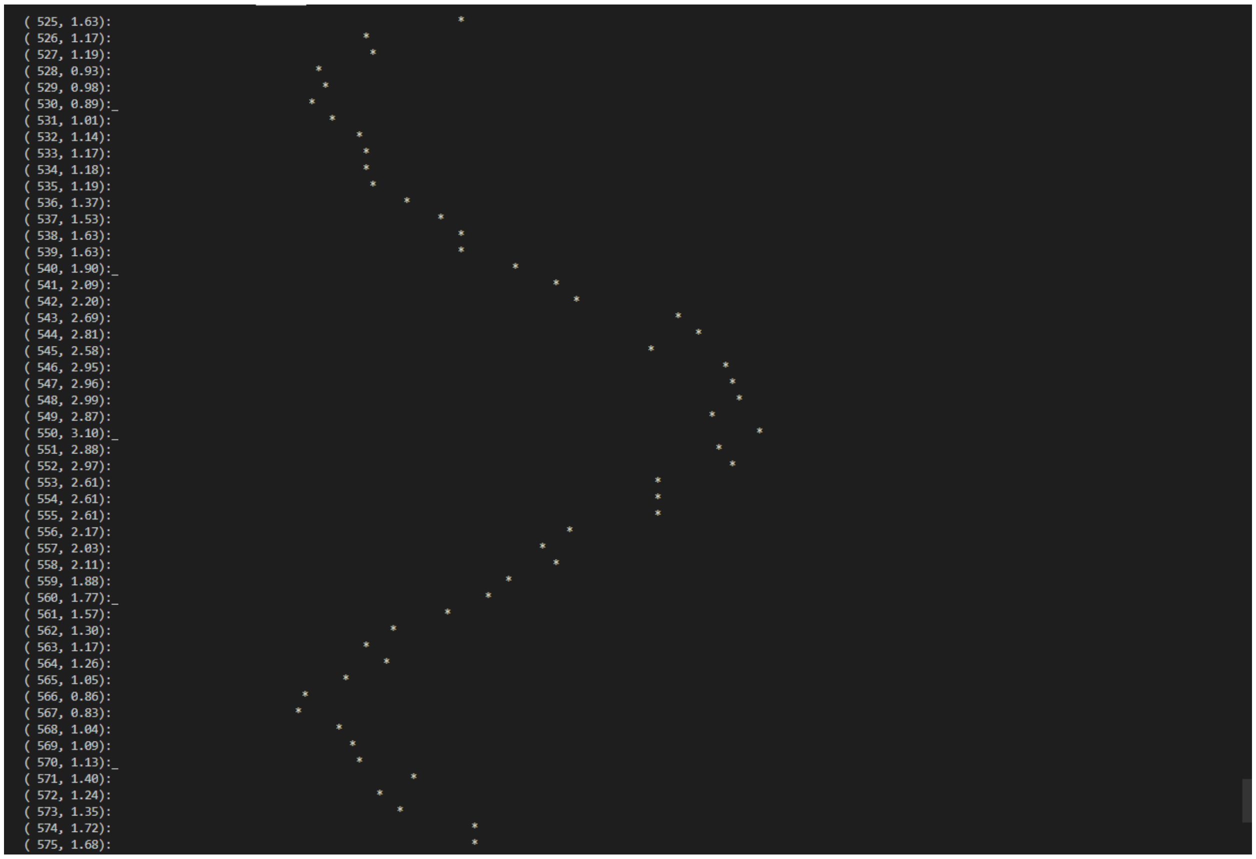
```
(1050, 0.74):_
(1051, 0.53):
(1052, 0.53):
(1053, 0.74):
(1054, 0.94):
(1055, 0.92):
(1056, 0.68):
(1057, 0.43):
(1058, 0.39):
(1059, 0.56):
(1060, 0.74):_
(1061, 0.69):
(1062, 0.44):
(1063, 0.18):
(1064, 0.14):
(1065, 0.32):
(1066, 0.51):
(1067, 0.49):
(1068, 0.26):
(1069, 0.03):
(1070, 0.03):_
(1071, 0.26):
(1072, 0.49):
(1073, 0.51):
(1074, 0.32):
(1075, 0.14):
(1076, 0.18):
(1077, 0.44):
(1078, 0.69):
(1079, 0.74):
(1080, 0.56):_
(1081, 0.39):
(1082, 0.43):
(1083, 0.68):
(1084, 0.92):
(1085, 0.94):
(1086, 0.74):
(1087, 0.53):
(1088, 0.53):
(1089, 0.74):
(1090, 0.94):_
```

rand() - 15/12/20

I then included the rand() function to create fluctuations in the function. The effect rand() has can be controlled by increasing or decreasing the modulus applied to the function.

```
/* sine_plotter.c */

/*Libraries*/
#include <stdio.h>
#include <stdint.h>
#include <math.h>

/*Definitions*/
#define SUCCESS 0
#define pi 3.14159265358979
#define MAX_TERMINAL 30
#define FREQ_A 10
#define FREQ_B 0
#define PHASE_A 0
#define PHASE_B 0
#define BACK_COL "\x1B[39m" /*Colour solution from David Guyon on
StackOverflow*/
#define FORE_COL "\x1B[49m" /*0=black, 1=red, 2=green, 3=yellow, 4=blue,
5=magenta, 6=cyan, 7=white*/
int plotval(float);
int rand(void);

/*Main code*/
int main()
{
    uint32_t x = 0;                                                /*Declare x as
fixed width unsigned variable, initialise equal to 0*/
    float y = 0;

    for (;;)                                                       /*Endless
loop*/
    {
        x++;                                                       /*Increase x by
1*/
        printf(BACK_COL FORE_COL "(%4d, ", x);                  /*Print value
of x to command line*/
        y = (sin(((FREQ_A * x) + PHASE_A) * pi / 180) +         /*calculate
value of y*/
             sin(((FREQ_B * x) + PHASE_B) * pi / 180) + 2) +
            ((((float)(rand() % 200) - 100) / 100) /5);
/*Random number used to create noise, from tutorialspoint*/
        printf("%0.2f):", y);                                   /*Print value
of y to command line*/
        if (x % 10 == 0)                                        /*If multiple
of 10 ad tick*/
        {
            printf("_");
        }
        else                                                    /*Otherwise add
space, so function is aligned*/
        {
            printf(" ");
        }
        plotval(y); /*Call Plotval*/
    }
    return SUCCESS;
}

/*plotval function*/
int plotval(float ploti) /*Declare plotval function and plot input*/
{
    for (int i = 0; i <= roundf(ploti * MAX_TERMINAL); i++) /*Loop till at
proportion of the screen to sin value*/
    {
        printf(" ");
    }
    printf("*\n"); /*Mark with an asterix*/
}
```

Owing to the large modulus value, the rand() function creates a large amount of fluctuation in the output waveform.



```
( 525, 1.63):                                                           *
( 526, 1.17):                                           *
( 527, 1.19):                                            *
( 528, 0.93):                                    *
( 529, 0.98):                                     *
( 530, 0.89):_                                 *
( 531, 1.01):                                      *
( 532, 1.14):                                          *
( 533, 1.17):                                           *
( 534, 1.18):                                           *
( 535, 1.19):                                            *
( 536, 1.37):                                                  *
( 537, 1.53):                                                        *
( 538, 1.63):                                                           *
( 539, 1.63):                                                           *
( 540, 1.90):_                                                                   *
( 541, 2.09):                                                                          *
( 542, 2.20):                                                                              *
( 543, 2.69):                                                                                               *
( 544, 2.81):                                                                                                  *
( 545, 2.58):                                                                                          *
( 546, 2.95):                                                                                                      *
( 547, 2.96):                                                                                                       *
( 548, 2.99):                                                                                                         *
( 549, 2.87):                                                                                                    *
( 550, 3.10):_                                                                                                           *
( 551, 2.88):                                                                                                     *
( 552, 2.97):                                                                                                       *
( 553, 2.61):                                                                                           *
( 554, 2.61):                                                                                           *
( 555, 2.61):                                                                                           *
( 556, 2.17):                                                                             *
( 557, 2.03):                                                                        *
( 558, 2.11):                                                                          *
( 559, 1.88):                                                                   *
( 560, 1.77):_                                                              *
( 561, 1.57):                                                         *
( 562, 1.30):                                                *
( 563, 1.17):                                           *
( 564, 1.26):                                               *
( 565, 1.05):                                        *
( 566, 0.86):                                 *
( 567, 0.83):                                *
( 568, 1.04):                                       *
( 569, 1.09):                                         *
( 570, 1.13):_                                         *
( 571, 1.40):                                                   *
( 572, 1.24):                                              *
( 573, 1.35):                                                 *
( 574, 1.72):                                                             *
( 575, 1.68):                                                             *
```

C3: Operators and Arrays
https://secure.ecs.soton.ac.uk/notes/ellabs/1/c3/c3.pdf
https://en.wikipedia.org/wiki/Floating-point_arithmetic
https://www.tutorialspoint.com/cprogramming/c_arrays.htm
https://stackoverflow.com/questions/4955198/what-does-dereferencing-a-pointer-mean/4955297
https://www.tutorialspoint.com/cprogramming/c_arrays.htm
https://stackoverflow.com/questions/4955198/what-does-dereferencing-a-pointer-mean/4955297

Preparation - 15/10/20

1) The (float) in front of RAND_MAX inside the rnd() function is called a type cast: it will turn the integer RAND_MAX into a floating point number before the division. Why is this necessary?

Inside rnd() the integer RAND_MAX must be turned into a float because the function returns a float.

2) The simulated evolution should stop when the best solution found yields a y-value no more than EPSILON away from the target value, or the maximum number of generations (MAX_GEN) has been reached. Write down the condition in the while-loop you will need for this.

best_ifit > EPSILON && gen < MAX_GEN

3) What part(s) of Darwin's algorithm (Reproduction, Variation, Selection) is happening in the for-loop inside main()?

Selection.

4) Write down the condition needed in the if-statement inside main().

ifit < best_ifit

5) Draft the definition for the declared initpop() function. It should initialize all positions in the array with random values. (The evolution starts with a population of random genes.)

```
//Sets up an a array with all positions a random number
void initpop(float *pop, int size)
{
    for (int i = 0; i < size; i++)
    {
        *(pop + i) = rnd();
    }
}
```

6) Draft the definition for the declared offspring() function. It should copy the parent to position 0 in the population array (this makes sure we always keep the best solution from the previous generation—a parent is immortal until a better child emerges). And further, it should copy the parent to all other positions in the array, but vary the offspring such created by adding or subtracting a random value. The argument mutst (mutation strength) should be used as a scaling factor for this variation.

```
//repopulates the array with values close to the best of the last generation
void offspring(float parent, float mutst, float *pop, int size)
{
    //parent is best value of last round
    *pop = parent;
    //fills the rest of the array with variations on the best value
    for (int i = 1; i < size; i++)
    {
        float rand_mut = (2*rnd() - 1)* mutst;
        *(pop + i) = parent + rand_mut;
    }
}
```

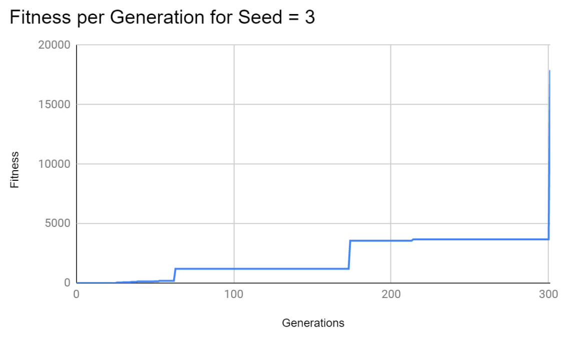
Quadratic Solver with Evolution - 21/10/20

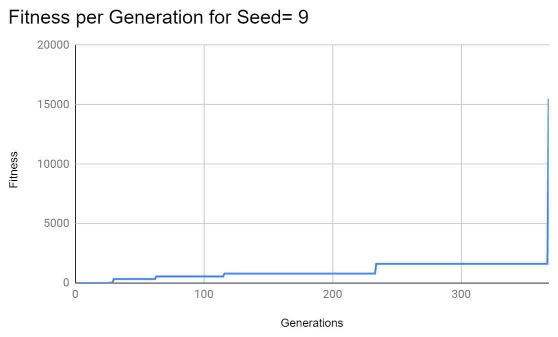
I have modified my equation to be $y = x^3 - 42$, I will run this equation with different seeds to find the value of the cube root of 42.

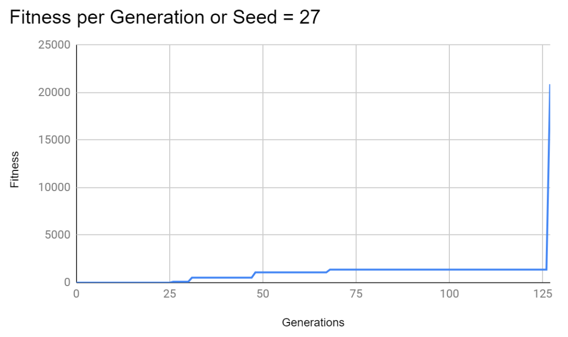
Run	Seed	Generations	Solutions
1	3	301	3.476025
2	9	368	3.476025
3	27	127	3.476028
4	81	280	3.476028
5	243	546	3.476027

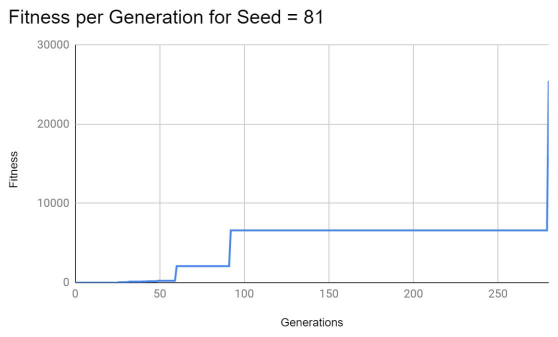
All the seeds will find the root within acceptable tolerance from the true value. I found that the value is between 3.476025 and 3.476028, with the true value being 3.4760266…. Meaning it is accurate to 6 significant figures.

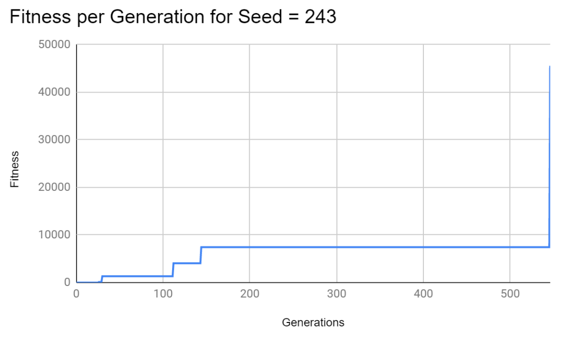
Saving Terminal Output to a File - 21/10/20







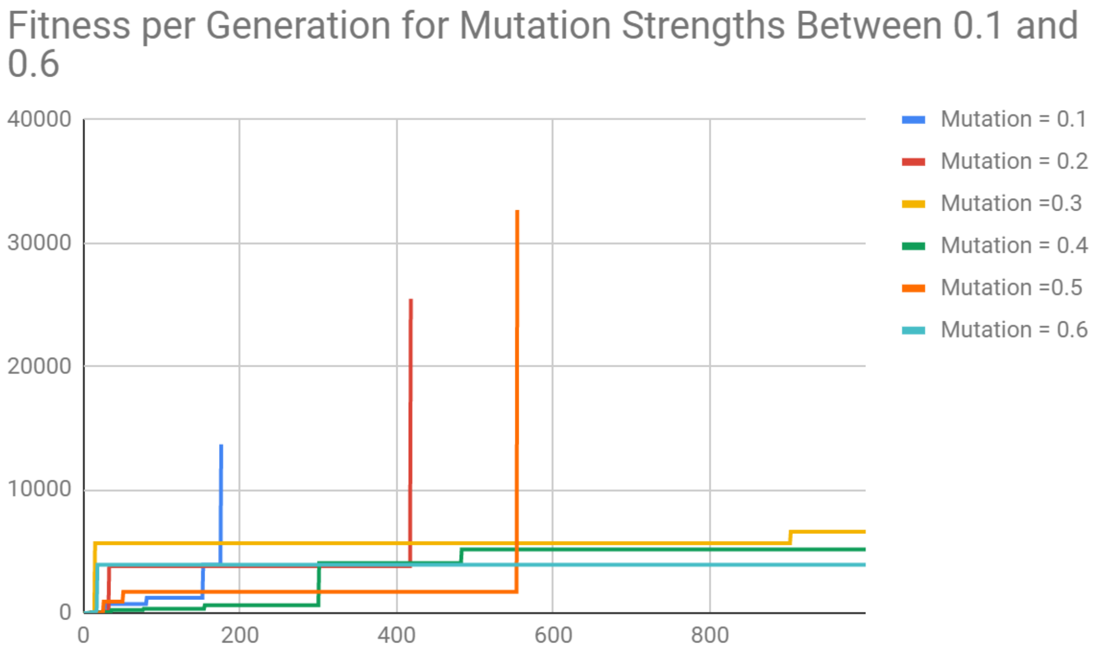




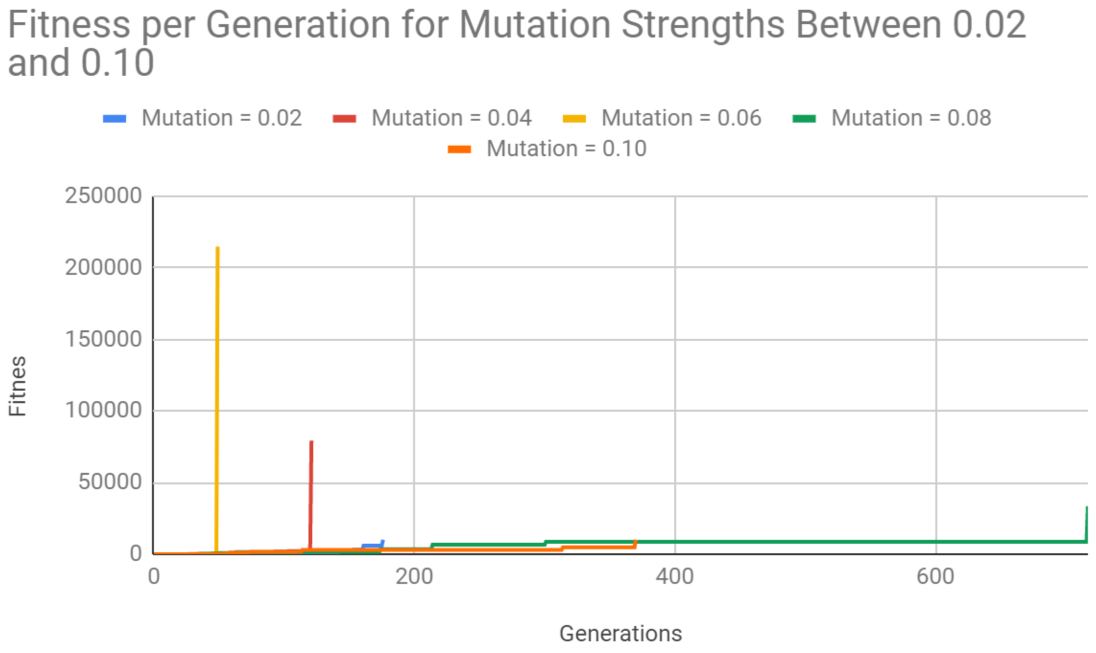
I found that there is a slow incremental increase for long periods. Then for small periods of a couple generations where massive improvements are made. I would guess this is because the programme gets stuck on a sub-optimum path for a long time and is waiting for a mutation to let it break through.

Investigating How Mutation Strength Affects the Number of Generations - 21/10/20

When investigating the effects mutation strength has on how long it takes the evolution program to find the (good enough) root I expect that too small a value then it will take long because it can only make incremental steps, but too large a value and it will fluctuate too much and will only be able to find the value by luck.

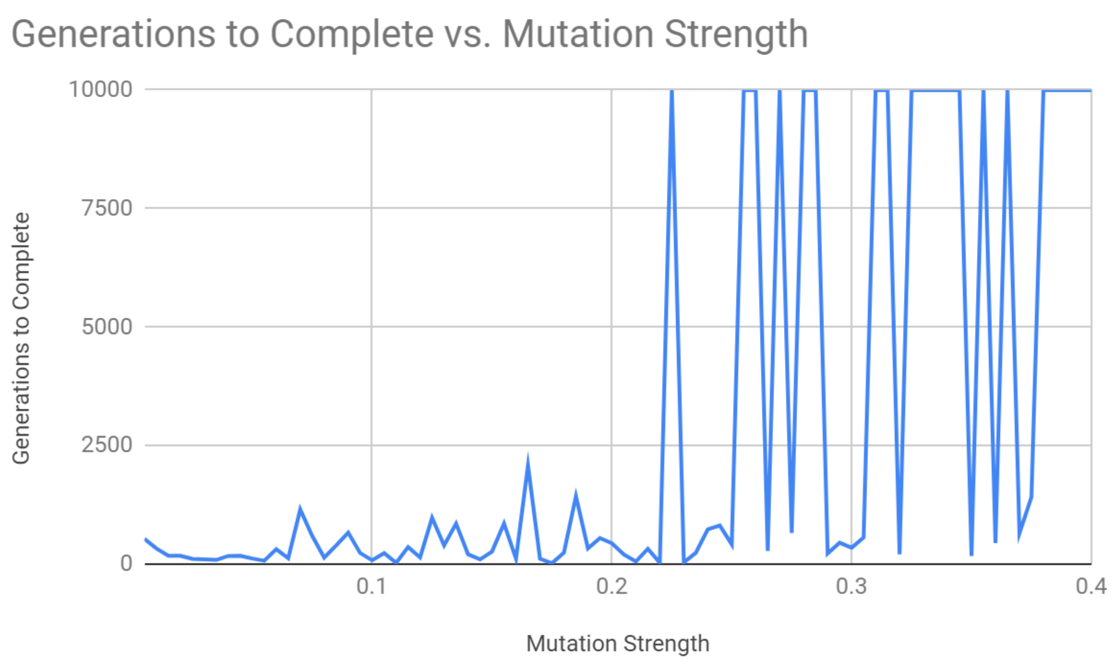


I found that for larger values of mutation strength it takes longer to find the value. I expect mutation strength = 0.5 is an outlier as it had a consistently low fitness and only found the value by chance.



I found that with the smaller values that the mutation strengths between 0.02 and 0.06 found their solutions much faster than the higher values. From this I would suggest that the mutation strength that takes the least generations would be around 0.05.

By editing my code I was able to make it print out the number of generations it took for each value of mutation strength.



With this program I tested a large sample size of mutation strengths and I found that a value below 0.2 would reliably find a solution. However above 0.2 the mutation Strength is high enough that it is unable to consistently find the root.

The Seceder Model of Evolution - 21/10/20

“We can take a different perspective on our population. Let’s assume the individuals in our population (the array of floats) are fashion-conscious and hungry for attention. Limit the possible characteristics (float value) of the individuals to be in the interval 0.0–1.0. If three of them meet, then whoever stands out most in the group wins. To do this we calculate the difference between any pair in the group of three, and sum for any individual in the group the differences. The individual with the largest sum wins the round. As a reward a (small) number of followers are created by copying the winning individual with small random variations (staying within the 0.0–1.0 bounds). These followers are placed back into the array at a random location (i.e. they replace random other individuals). What do you expect to happen if we repeat this process indefinitely? “

In this scenario I anticipate that the array will homogenise onto one or two of the fittest values. There will be small variation on this (or these) values that could lead to small increases in fitness but I anticipate as time goes on these will become rare. If one of these groups gets enough incremental increases then in a few generations it will wipe out the other group.

C4: Pointers, Files and Strings
https://www.tutorialspoint.com/cprogramming/c_pointers.htm#:~:text=A%20pointer%20is%20a%20variable,to%20store%20any%20variable%20address.
https://beginnersbook.com/2014/01/c-pointers/
https://www.programiz.com/c-programming/library-function/ctype.h/isalpha

Preparation - 24/10/20

1) Describe the behaviour of the * and & operators in C with regard to pointers.

In C the * operator (in regards to pointers) is used to turn a memory address into a variable.

In C the & operator (in regards to pointers) is used to return the address of a piece of information in memory.

2) Detail how an integer array can be passed as an argument to a function by reference so that the function is able to update the contents of the array and return these changes to the calling function.

An integer array can be passed as an argument to a function by giving the first address of the array as a pointer, like so:

```
void func(int *point, array_size);
```

The function can then use a for loop to update every position in the array, like so:

```
for (int i = 0; i < array_size; i++)
    {
        *(pop + i) = rand();
    }
```

3) Look up the definitions for FILE, fopen, fclose, fgetc and EOF and describe their purpose.

FILE is a data type to asigns files from another location to memory.

fopen() is a function that gets you a file pointer.

fclose() is a function that clears the memory of information from the file pointer.

fgetc() is a function that gets a character from a file, and moves the position in the file along one.

EOF stands for end of file and is an ASCII character (hex value: 0x00) that signifies the end of a file's information.

4) Write code that opens a text file for reading and displays the contents of the file on stdout.

```c
#include <stdio.h>
#include <ctype.h>

int main()
{
    FILE *fp;
    fp = fopen("lazy_doggo.txt", "r"); //Opens file for reading

    for (;;)
    {
        if (feof(fp)) //Makes sure it is not end of the file
        {
            break;
        }
        printf("%c", fgetc(fp)); //reads value from file, prints value, and
position along
    }
    fclose(fp);
}
```

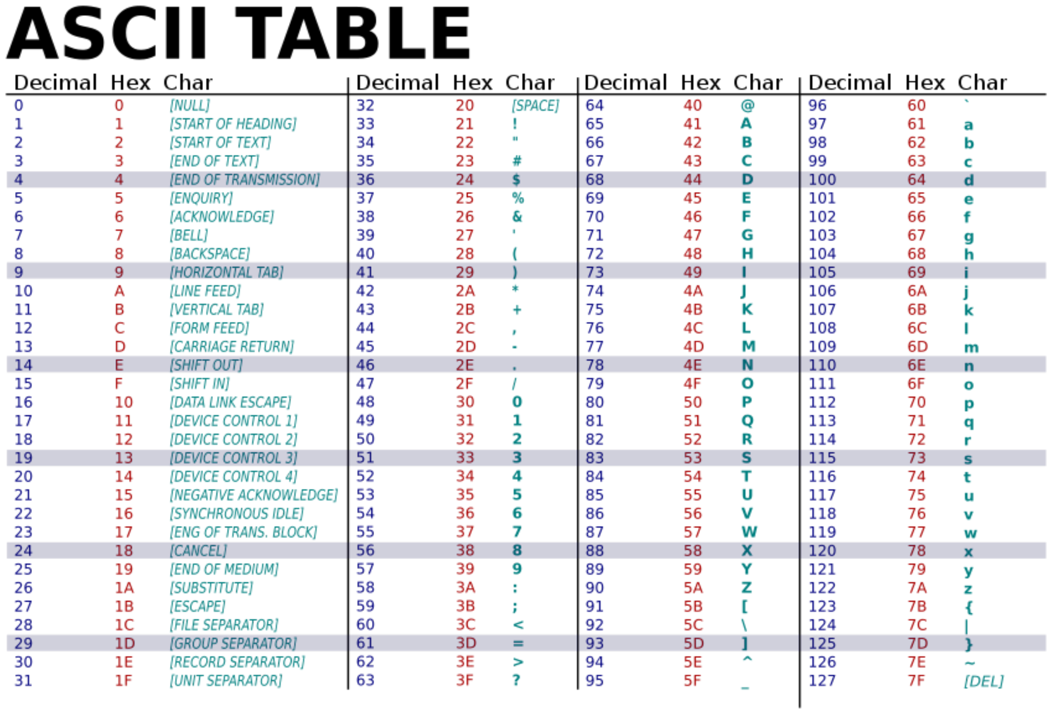
5) Which header file contains the interfaces for isalpha and toupper? Describe the operation of these two library functions.

isalpha() and toupper() are from the <ctype.h> library, a library for character processing.

isalpha() is a function that defines what type of character is read.
isalpha() == 0; means it is a non-alphabetic character
isalpha() == 1; means it is an uppercase letter
isalpha() == 2; means it is a lowercase letter

toupper() is a functions that changes a lowercase letter into an uppercase letter

6) Characters in C are represented in ASCII format. What value is stored in memory to represent the character ’A’?



ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

7) Look up the operation of a Caesar Cipher and describe how it works.

A caesar cipher is a cipher that shifts values by a set amount (traditionally by three places to the left). It is usually attributed to being invented by Julius Caesar, hence the name.

8) What operator is used to implement the modulo operation in C?

In C the % operator (in regards to pointers) is used to return the modulus of a value.

Reading From Files - 28/10/20

By including the correct library and using toupper() I managed to make my program read a text file and output it to terminal in caps.

```
for (;;)
    {
        if (feof(fp)) //Makes sure it is not end of the file
        {
            break;
        }
        printf("%c", toupper(fgetc(fp))); //reads value from file, prints
value, and position along
    }
```

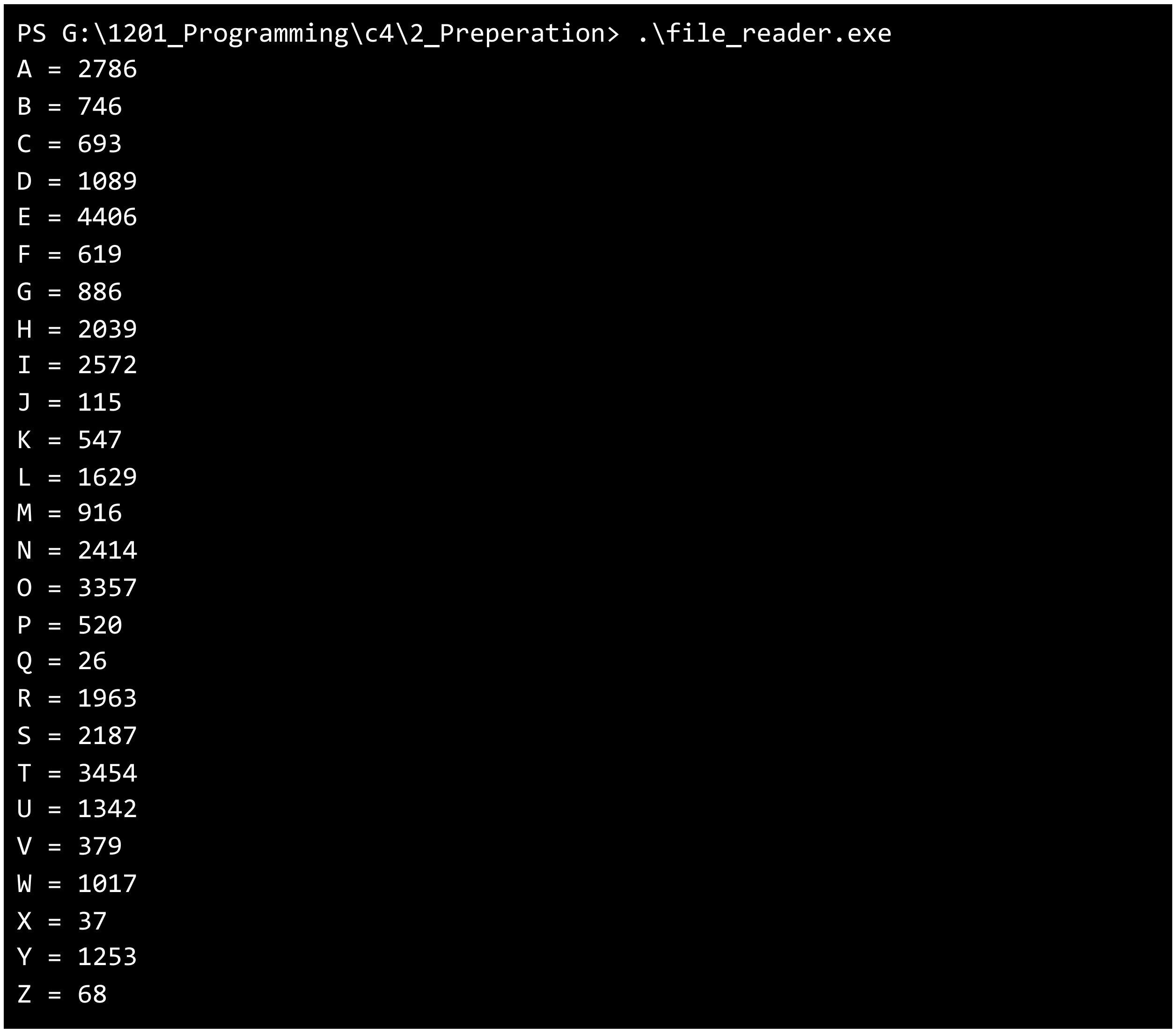
```
PS G:\1201_Programming\c4\2_Preperation> gcc file_reader_back001.c -o
file_reader
PS G:\1201_Programming\c4\2_Preperation> .\file_reader.exe
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG.ƒ
```

I then created a function that incremented the value in a frequency array for every new letter to find the values of the histogram.

```c
void calculateHistogram(const char *file, int *point)
{
    fp = fopen(file, "r"); //Opens file for reading

    for (;;)
    {
        if (feof(fp)) //Makes sure it is not end of the file
        {
            break;
        }
        char letter = toupper(fgetc(fp)); //reads value from file, prints 
value, and position along
        if (isalpha(letter) != 1)         //makes sure is a letter
        {
            continue;
        }
        *(point + (int)letter - 65) += 1; //add one to array position for 
letter
    }
    fclose(fp);
}
```

I then created a print function that printed every letter followed by the frequency it occurs in the array. I achieve this by passing in the frequency array and printing the value.

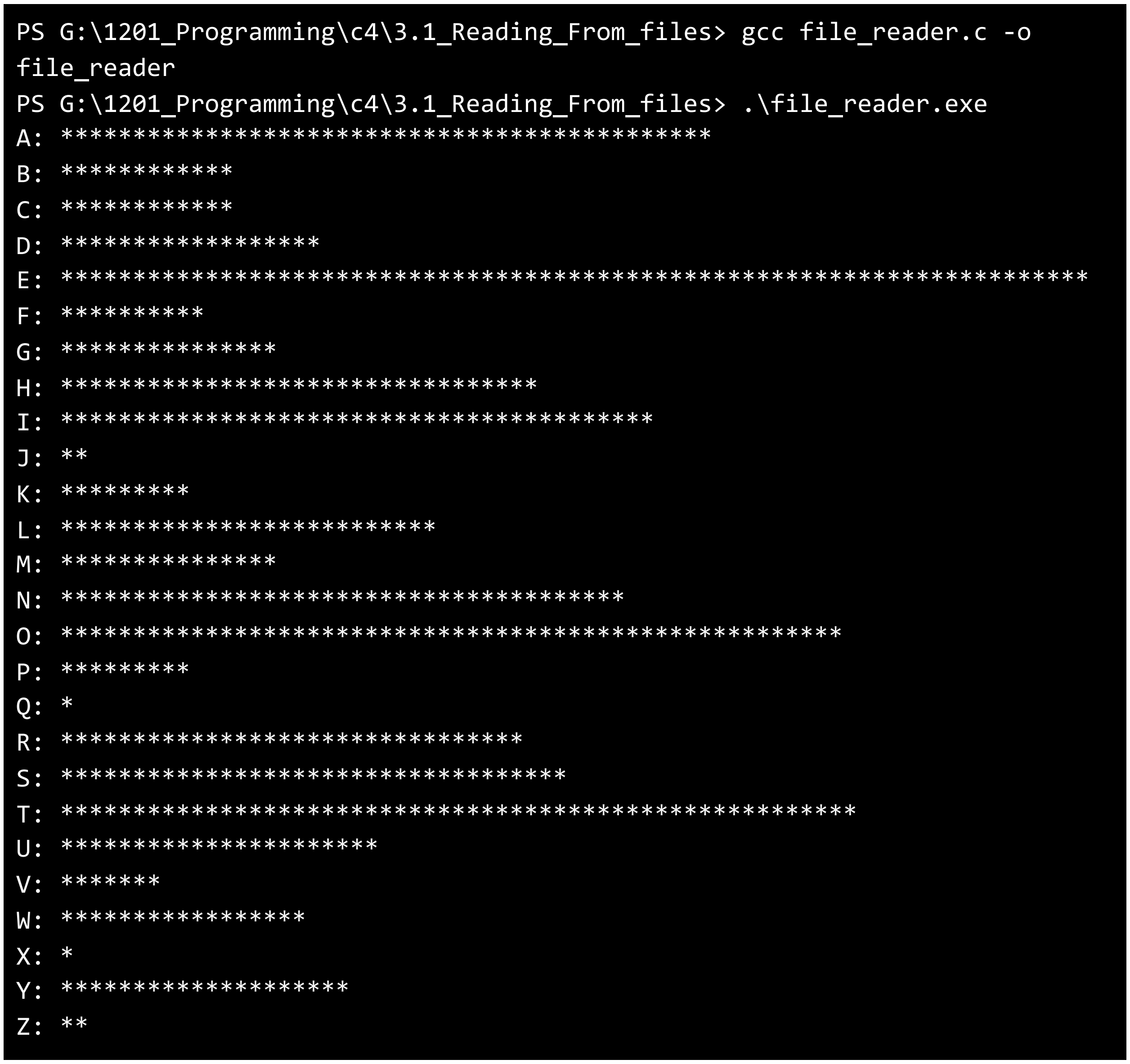
```c
void printHistogram(const int array[])
{
    for (int i = 0; i < alpha_array; i++)
    {
        printf("%c = %i\n", (char)(i+65), *(frequency + i));
    }
}
```



```
PS G:\1201_Programming\c4\2_Preperation> .\file_reader.exe
A = 2786
B = 746
C = 693
D = 1089
E = 4406
F = 619
G = 886
H = 2039
I = 2572
J = 115
K = 547
L = 1629
M = 916
N = 2414
O = 3357
P = 520
Q = 26
R = 1963
S = 2187
T = 3454
U = 1342
V = 379
W = 1017
X = 37
Y = 1253
Z = 68
```

I then replace this with a function that graphs the values across the terminal. First I tested every value in the frequency array to find its maximum value. After knowing the furthest it can go I used a for loop, similar to the one used for my sine plotter to plot the bars of my histogram

```c
void graphHistogram(const int array[])
{
    int max_array = 1;
    for (int i = 0; i < alpha_array; i++)
    {
        if (frequency[i] > max_array) // If the frequency is larger than any of the previous update the largest value
        {
            max_array = frequency[i];
        }
    }

    for (int i = 0; i < alpha_array; i++) // For every letter print the letter, histogram bar, and move to new line
    {
        printf("%c: ", (char)(i + 65));
        for (int j = 0; j <= (frequency[i] / (float)max_array) * MAX_TERMINAL; j++) // Loop till at proportion of the screen for histogram
        {
            printf("*");
        }
        printf("\n");
    }
}
```



```
PS G:\1201_Programming\c4\3.1_Reading_From_files> gcc file_reader.c -o
file_reader
PS G:\1201_Programming\c4\3.1_Reading_From_files> .\file_reader.exe
A: *********************************************
B: ************
C: ************
D: ******************
E: ***********************************************************************
F: **********
G: ***************
H: *********************************
I: *****************************************
J: **
K: *********
L: **************************
M: ***************
N: ***************************************
O: ******************************************************
P: *********
Q: *
R: ********************************
S: ***********************************
T: *******************************************************
U: **********************
V: *******
W: *****************
X: *
Y: ********************
Z: **
```

Manipulating Strings - 28/10/20

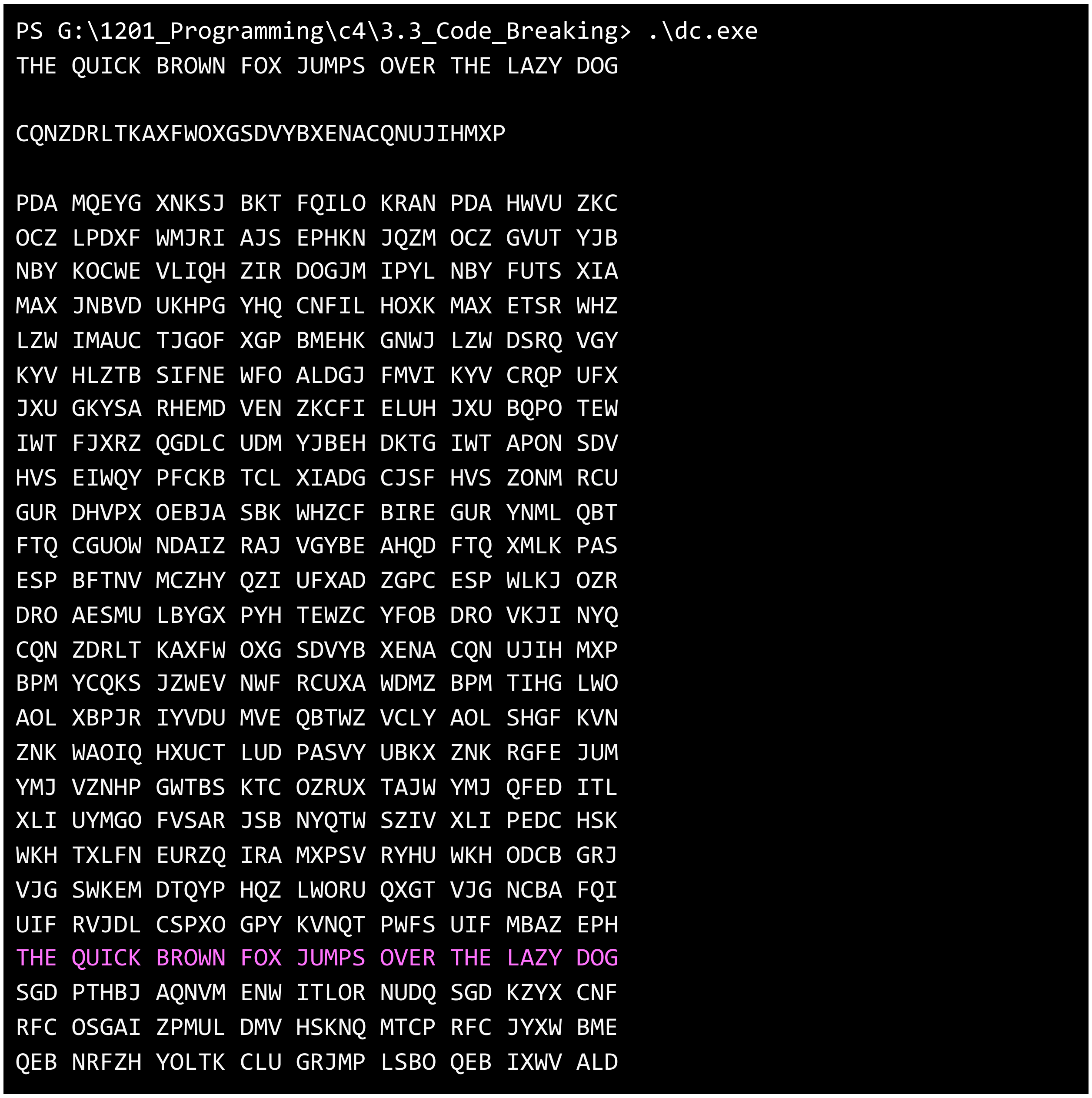
Using the modulo operator I created code that could encipher and decipher a message given a fixed offset. This applied and undid a caesar cipher of a fixed length.

```c
void encipher(const char *p, char *c, const unsigned int offset)
{
    for (int i = 0; i < strlen(p); i++)
    {
        if (isalpha(p[i]) != 1) //makes sure is a letter
        {
            c[i] = p[i];
            continue;
        }
        c[i] = (p[i] + offset) % 26 + 65;
        printf("%c", c[i]);
    }
    printf("\n\n");
}

void decipher(const char *c, char *p, const unsigned int offset)
{
    for (int i = 0; i < strlen(c); i++)
    {
        if (isalpha(c[i]) != 1) //makes sure is a letter
        {
            p[i] = c[i];
            printf("%c", p[i]);
            continue;
        }
        p[i] = (c[i] - offset) % 26 + 65;
        printf("%c", p[i]);
    }
    printf("\n");
}
```

```
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

XLIUYMGOFVSARJSBNYQTWSZIVXLIPEDCHSK

THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
```

I was able to modify my code so that it could brute force all 26 possible versions of a random offset.

```c
int main()
{
    FILE *fp;
    fp = fopen("lazy_doggo.txt", "r"); //Opens file for reading
    char plain[max_text];
    char coded[max_text];
    srand(3);
    int offset = rand() % 26;
    int array = 26;

    for (int i = 0;; i++)
    {
        if (feof(fp)) //Makes sure it is not end of the file
        {
            break;
        }
        char text = toupper(fgetc(fp));
        plain[i] = (isalpha(text) == 1) ? text : ' ';
        printf("%c", plain[i]); //reads value from file, prints value, and
position along
    }
    fclose(fp);
    printf("\n\n");

    encipher(plain, coded, offset);
    for (int i = 0; i < array; i++)
    {
        decipher(coded, plain, i);
    }
}
```



```
PS G:\1201_Programming\c4\3.3_Code_Breaking> .\dc.exe
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

CQNZDRLTKAXFWOXGSDVYBXENACQNUJIHMXP

PDA MQEYG XNKSJ BKT FQILO KRAN PDA HWVU ZKC
OCZ LPDXF WMJRI AJS EPHKN JQZM OCZ GVUT YJB
NBY KOCWE VLIQH ZIR DOGJM IPYL NBY FUTS XIA
MAX JNBVD UKHPG YHQ CNFIL HOXK MAX ETSR WHZ
LZW IMAUC TJGOF XGP BMEHK GNWJ LZW DSRQ VGY
KYV HLZTB SIFNE WFO ALDGJ FMVI KYV CRQP UFX
JXU GKYSA RHEMD VEN ZKCFI ELUH JXU BQPO TEW
IWT FJXRZ QGDLC UDM YJBEH DKTG IWT APON SDV
HVS EIWQY PFCKB TCL XIADG CJSF HVS ZONM RCU
GUR DHVPX OEBJA SBK WHZCF BIRE GUR YNML QBT
FTQ CGUOW NDAIZ RAJ VGYBE AHQD FTQ XMLK PAS
ESP BFTNV MCZHY QZI UFXAD ZGPC ESP WLKJ OZR
DRO AESMU LBYGX PYH TEWZC YFOB DRO VKJI NYQ
CQN ZDRLT KAXFW OXG SDVYB XENA CQN UJIH MXP
BPM YCQKS JZWEV NWF RCUXA WDMZ BPM TIHG LWO
AOL XBPJR IYVDU MVE QBTWZ VCLY AOL SHGF KVN
ZNK WAOIQ HXUCT LUD PASVY UBKX ZNK RGFE JUM
YMJ VZNHP GWTBS KTC OZRUX TAJW YMJ QFED ITL
XLI UYMGO FVSAR JSB NYQTW SZIV XLI PEDC HSK
WKH TXLFN EURZQ IRA MXPSV RYHU WKH ODCB GRJ
VJG SWKEM DTQYP HQZ LWORU QXGT VJG NCBA FQI
UIF RVJDL CSPXO GPY KVNQT PWFS UIF MBAZ EPH
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
SGD PTHBJ AQNVM ENW ITLOR NUDQ SGD KZYX CNF
RFC OSGAI ZPMUL DMV HSKNQ MTCP RFC JYXW BME
QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD
```

C5: Data Structures and Dynamic Memory Allocation

Preparation - 02/11/20

1) Describe the concepts of a struct and an enum in C.

A struct{} is a collection of variables.
A enum{} assigns names to a set of integers.

2) By inspecting the source code in vector.c determine the appropriate data structure definition to go into vector.h.

```c
typedef struct {
    /* TODO */
    int length;
    double* element;
} Vector;
```

3) By inspecting the source code in matrix.c determine the appropriate data structure definition to go into matrix.h.

```c
typedef struct {
    /* TODO */
    int rows;
    int cols;
    double** element;
} Matrix;
```

4) Describe the operation of the malloc and calloc functions, commenting on any differences.

malloc() requests memory from the operating system and returns it to the program.
calloc() requests memory, assigns zeros to all the positions and returns it to the program.

5) How can you give back memory that you have dynamically allocated?

free() gives back memory to the operating system that had been dynamically assigned to the program.

6) How do you compile a project with multiple source files?

Using the tool make (from MinGW) and a makefile with several terminal commands, you can compile a project from multiple source files.

7) How can arguments be passed from the command line into your program?

```
int main(int argc, char* argv[])
```

argc is the number of strings on the command line (the first string is always the program name). argv is a vector of pointers to character strings (argv[0] is the address of the string containing the program's name).

Vectors - 04/11/20

Having defined the data structure for a vector in <vector.h> for my preparation I created a main function to call the other functions. I also wrote the createVector() function that dynamically assigns memory for the vector. After this I ran the code and it read the vectors from the example file.

```
int main()
{
    Vector v = createVectorFromFile("example1.vec");
    printVector(v);
    destroyVector(v);
}

Vector createVector(const unsigned int nLength)
{
    /* TODO */
    Vector vec = {nLength, NULL};
    vec.element = (double *)calloc(nLength, sizeof(int));
    return vec;
}
```

```
PS G:\1201_Programming\c5\3.1_Vectors> .\vector.exe
[0] = 2.300000
[1] = 4.500002
```

Matrices - 04/11/20

Having defined the data structure for a matrix in <matrix.h> for my preparation I created a main function to call the other functions. I also wrote the createMatrix() function that dynamically assigns memory for the matrix. After this I ran the code and it read the matrices from the example file.

```c
int main()
{
    Matrix m = createMatrixFromFile("example1.mat");
    printMatrix(m);
    destroyMatrix(m);
}

Matrix createMatrix(const unsigned int nRows, const unsigned int nCols)
{
    /* TODO */
    Matrix mat = {nRows, nCols, NULL};
    mat.element = (double **)calloc(nCols, sizeof(int));
    for (int i = 0; i < nCols; i++)
    {
        mat.element[i] = calloc(nRows, sizeof(int));
    }

    return mat;
}
```

```
PS G:\1201_Programming\c5\3.2_Matrices> .\matrix.exe
[0][0] = 2.300000
[0][1] = 4.500000
[1][0] = 2.400000
[1][1] = 6.300002
```

Circuit Simulation - 04/11/20

In <circuit.h> I created the structure definitions for all of my data types. I was able to ascertain what each structure needed by looking at what occurred in the provided code of circuit.c.

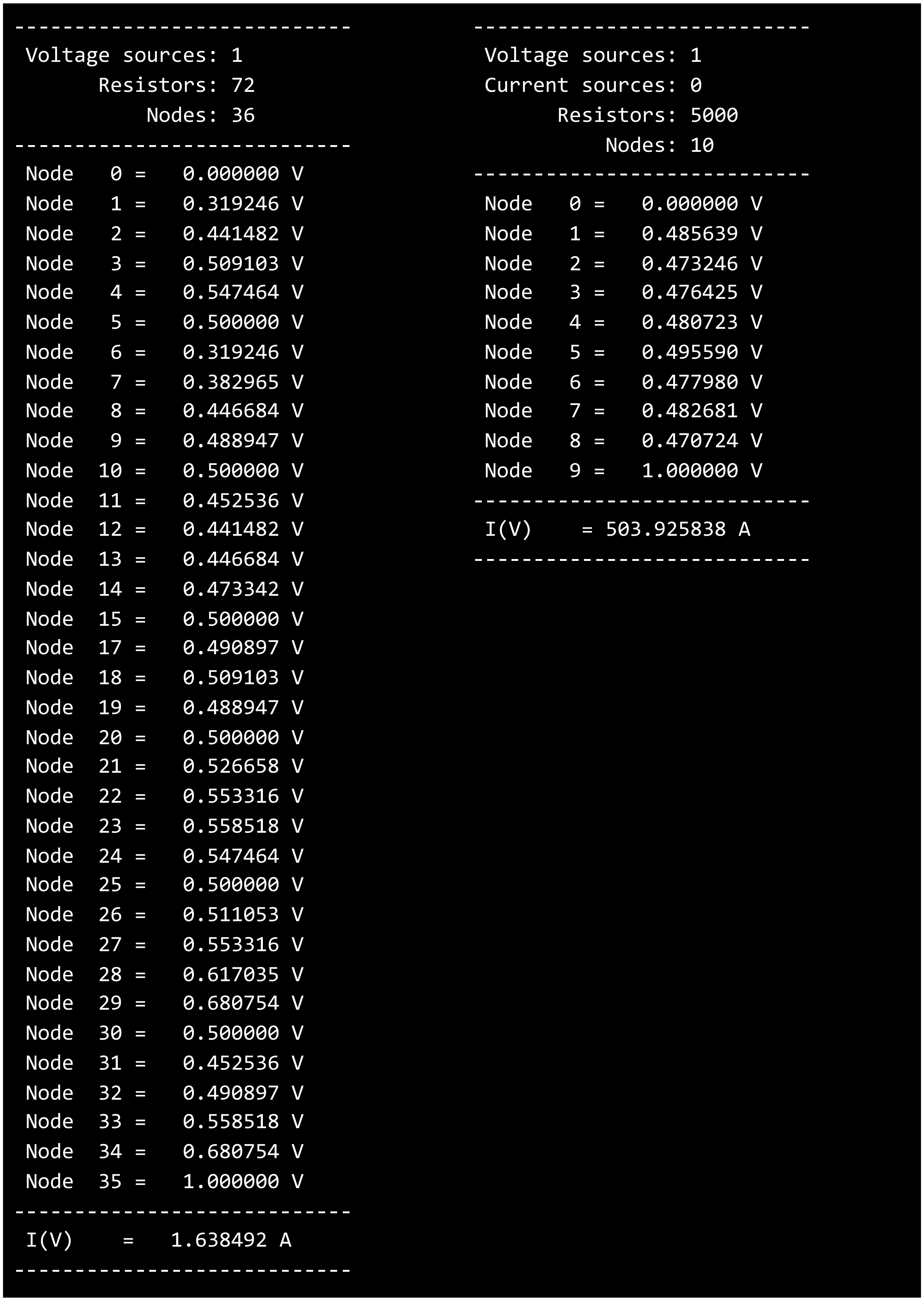
```
/* Data Structures */

typedef enum {
/* TODO */
resistor,
voltage,
current
} CompType;

typedef struct {
/* TODO */
CompType type;
char name [32];
unsigned int n1;
unsigned int n2;
double value;
} Component;

typedef struct {
/* TODO */
int nV;
int nI;
int nR;
int nN;
int nC;
Component* comp;
} Circuit;
```

I then used the makefile to compile all my files together so that they would run as one program.

```
int main(int argc, char *argv[])
{
    Circuit c;

    if (argc == 2)
    {
        /* TODO */
        c = createCircuitFromFile(argv[1]);
        analyseCircuit(c);
        destroyCircuit(c);
        return EXIT_SUCCESS;
    }
    else
    {
        printf("Syntax: %s <filename>\n", argv[0]);
    }
    return EXIT_SUCCESS;
}
```

```
PS
G:\1201_Programming\c5\3.3_Circuit>
.\analyse .\example1.cir
----------------------------
 Voltage sources: 1
 Current sources: 0
       Resistors: 3
           Nodes: 3
----------------------------
 Node   0 =   0.000000 V
 Node   1 = -12.000000 V
 Node   2 =  -6.000000 V
----------------------------
 I(V1)    =   0.006000 A
----------------------------
```

```
PS
G:\1201_Programming\c5\3.3_Circuit>
.\analyse .\example2.cir
----------------------------
 Voltage sources: 1
 Current sources: 1
       Resistors: 6
           Nodes: 5
----------------------------
 Node   0 =   0.000000 V
 Node   1 = 33611.548447 V
 Node   2 = 23111.111111 V
 Node   3 = 23110.111111 V
 Node   4 = 33607.049759 V
----------------------------
 I(V1)    =   1.499729 A
----------------------------
```

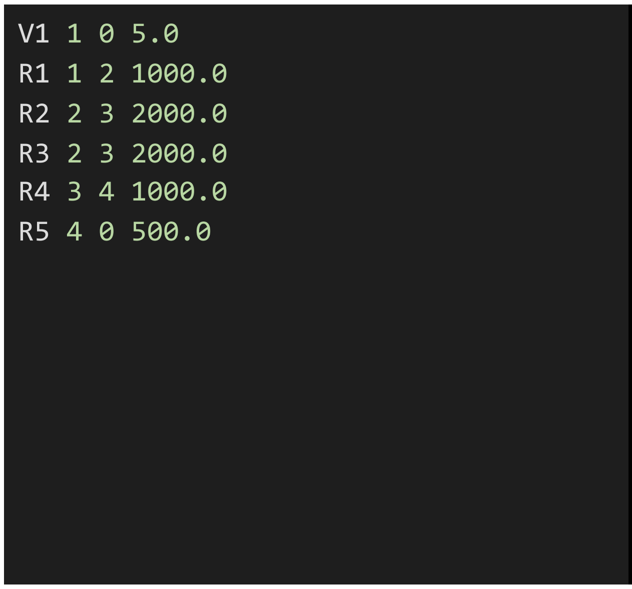


```
----------------------------
 Voltage sources: 1
       Resistors: 72
           Nodes: 36
----------------------------
 Node   0 =   0.000000 V
 Node   1 =   0.319246 V
 Node   2 =   0.441482 V
 Node   3 =   0.509103 V
 Node   4 =   0.547464 V
 Node   5 =   0.500000 V
 Node   6 =   0.319246 V
 Node   7 =   0.382965 V
 Node   8 =   0.446684 V
 Node   9 =   0.488947 V
 Node  10 =   0.500000 V
 Node  11 =   0.452536 V
 Node  12 =   0.441482 V
 Node  13 =   0.446684 V
 Node  14 =   0.473342 V
 Node  15 =   0.500000 V
 Node  17 =   0.490897 V
 Node  18 =   0.509103 V
 Node  19 =   0.488947 V
 Node  20 =   0.500000 V
 Node  21 =   0.526658 V
 Node  22 =   0.553316 V
 Node  23 =   0.558518 V
 Node  24 =   0.547464 V
 Node  25 =   0.500000 V
 Node  26 =   0.511053 V
 Node  27 =   0.553316 V
 Node  28 =   0.617035 V
 Node  29 =   0.680754 V
 Node  30 =   0.500000 V
 Node  31 =   0.452536 V
 Node  32 =   0.490897 V
 Node  33 =   0.558518 V
 Node  34 =   0.680754 V
 Node  35 =   1.000000 V
----------------------------
 I(V)    =   1.638492 A
----------------------------
```

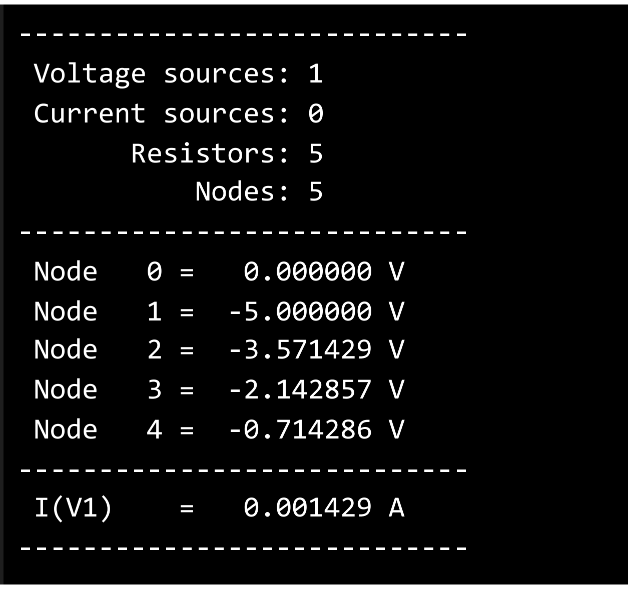
```
----------------------------
 Voltage sources: 1
 Current sources: 0
       Resistors: 5000
           Nodes: 10
----------------------------
 Node   0 =   0.000000 V
 Node   1 =   0.485639 V
 Node   2 =   0.473246 V
 Node   3 =   0.476425 V
 Node   4 =   0.480723 V
 Node   5 =   0.495590 V
 Node   6 =   0.477980 V
 Node   7 =   0.482681 V
 Node   8 =   0.470724 V
 Node   9 =   1.000000 V
----------------------------
 I(V)    = 503.925838 A
----------------------------
```

Netlists - 04/11/20

I was able to write a netlist in notepad++ having different voltage sources and resistor values by connecting the nodes together.

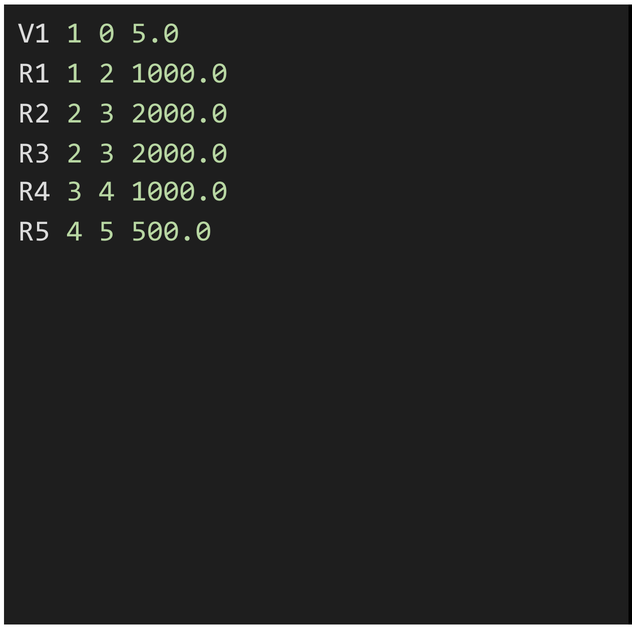


```
V1 1 0 5.0
R1 1 2 1000.0
R2 2 3 2000.0
R3 2 3 2000.0
R4 3 4 1000.0
R5 4 0 500.0
```

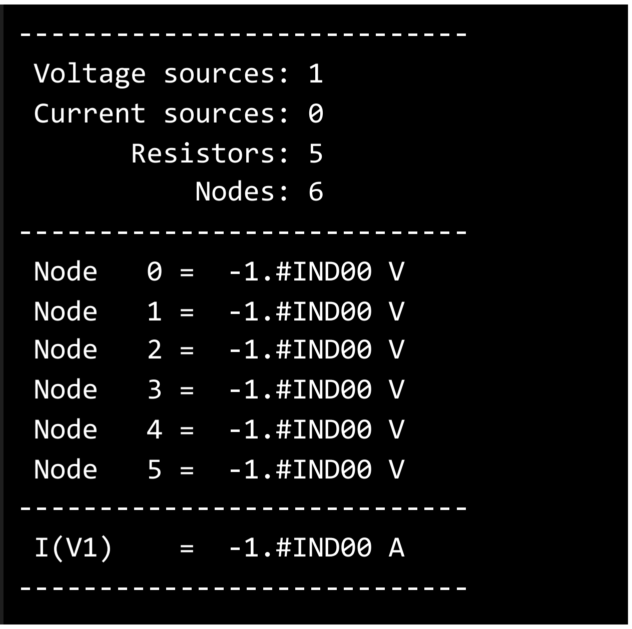


```
----------------------------
 Voltage sources: 1
 Current sources: 0
       Resistors: 5
           Nodes: 5
----------------------------
 Node   0 =   0.000000 V
 Node   1 =  -5.000000 V
 Node   2 =  -3.571429 V
 Node   3 =  -2.142857 V
 Node   4 =  -0.714286 V
----------------------------
 I(V1)    =   0.001429 A
----------------------------
```

In my next circuit I was able to break the program by not connecting all the nodes together and causing a divide by zero error in the programme. This stopped it from being able to correctly calculate the voltage over the resistors.



```
V1 1 0 5.0
R1 1 2 1000.0
R2 2 3 2000.0
R3 2 3 2000.0
R4 3 4 1000.0
R5 4 5 500.0
```



```
----------------------------
 Voltage sources: 1
 Current sources: 0
       Resistors: 5
           Nodes: 6
----------------------------
 Node   0 =  -1.#IND00 V
 Node   1 =  -1.#IND00 V
 Node   2 =  -1.#IND00 V
 Node   3 =  -1.#IND00 V
 Node   4 =  -1.#IND00 V
 Node   5 =  -1.#IND00 V
----------------------------
 I(V1)    =  -1.#IND00 A
----------------------------
```

Problem Sheet 1 - 03/01/21

1) Discuss the difference, advantages, and disadvantages of passing an argument to a function "by value" rather than "by reference".

Passing an argument "by reference" means substituting the value with a defined variable.
Passing an argument "by value" means using the set value.

Passing an argument by value is useful if the you are unlikely to need to change the value or if you are only using it once.

Passing by reference is useful as it allows you to easily change a portion of code without the chance of breaking another part of code.

2) To understand the error messages emitted by the compilation tool-chain, it is necessary to understand whether a message was raised by the Preprocessor, the Compiler or the Linker. Give for each of these three stages some examples of the types of errors that the stage will flag up.

Preprocessor: Text substitutions
An error in the preprocessor will occur if the program is missing a defined value.
Compiler: Translation of source code into assembly language
Assembler: Translation into machine instructions
An error in the compiler or assembler will occur if a command is used incorrectly.
Linker: Combination of translation units into executable code
An error in the Linker will occur if variables are passed between functions or files incorrectly.

3) What will be the output of the following program?

```c
#include <stdio.h>

int main()
{
    int a = 0;
    int b = 1;
    int c = 2;

    if (a != b) //if a and b are different
    {
        if (a) //if a is non zero print A
        {
            printf("A");
        }
        //set a to the value of b and if a is now non zero print A
        else if (a = b)
        {
            printf("B");
        }
        else               //otherwise print C
        {
            printf("C");
        }
    }
    if (a && c) //if a and c are both non zero then print AC
    {
        printf("AC");
    }
    else    //otherwise print !
    {
        printf("!");
    }
    return 0;
}
```

The output for this program will be BAC as the value of a will be updated half way through the program.

Problem Sheet 2 - 04/01/21

1) Using the laws of conservation of kinetic energy

$$\tfrac{1}{2}m_1v_1^2 + \tfrac{1}{2}m_2v_2^2 = \tfrac{1}{2}m_1u_1^2 + \tfrac{1}{2}m_2u_2^2$$

and conservation of momentum

$$m_1v_1 + m_2v_2 = m_1u_1 + m_2u_2$$

show that the velocities of the blocks after collision with each other are given by

$$v_1 = \frac{(m_1-m_2)u_1+2m_2u_2}{m_1+m_2}$$

$$v_2 = \frac{(m_2-m_1)u_2+2m_1u_1}{m_1+m_2}$$

where u_1 and v_1 are the velocities before and after collision respectively.

$$m_1v_1 + m_2v_2 = m_1u_1 + m_2u_2$$
$$m_2v_2 = m_1u_1 + m_2u_2 - m_1v_1$$
$$m_2v_2 = m_1(u_1 - v_1) + m_2u_2$$
$$v_2 = \frac{m_1(u_1-v_1)+m_2u_2}{m_2}$$

$$\tfrac{1}{2}m_1v_1^2 + \tfrac{1}{2}m_2v_2^2 = \tfrac{1}{2}m_1u_1^2 + \tfrac{1}{2}m_2u_2^2$$
$$m_1v_1^2 + m_2v_2^2 = m_1u_1^2 + m_2u_2^2$$
$$m_2v_2^2 = m_1u_1^2 + m_2u_2^2 - m_1v_1^2$$
$$m_2v_2^2 = m_1(u_1^2 - v_1^2) + m_2u_2^2$$
$$v_2^2 = \frac{m_1(u_1^2-v_1^2) + m_2u_2^2}{m_2}$$

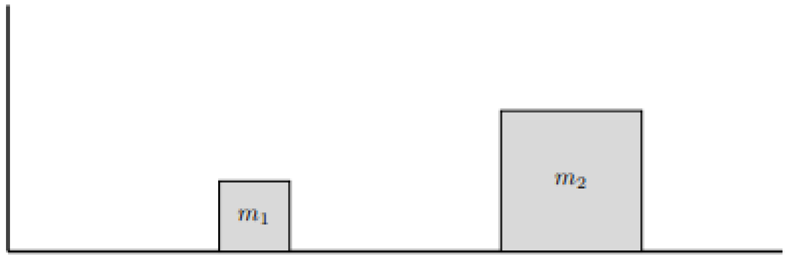
$$\left(\frac{m_1(u_1-v_1)+m_2u_2}{m_2}\right)^2 = \frac{m_1(u_1^2-v_1^2) + m_2u_2^2}{m_2}$$
$$\frac{m_1^2(u_1-v_1)^2 + 2m_1m_2u_2(u_1-v_1)^2 + m_2^2u_2^2}{m_2^2} = \frac{m_1(u_1^2-v_1^2) + m_2u_2^2}{m_2}$$
$$m_1^2(u_1 - v_1)^2 + 2m_1m_2u_2(u_1 - v_1) + m_2^2u_2^2 = m_1m_2(u_1^2 - v_1^2) + m_2^2u_2^2$$
$$m_1^2(u_1 - v_1)^2 + 2m_1m_2u_2(u_1 - v_1) = m_1m_2(u_1^2 - v_1^2)$$
$$m_1^2(u_1 - v_1)^2 + 2m_1m_2u_2(u_1 - v_1) = m_1m_2(u_1 - v_1)(u_1 + v_1)$$
$$m_1(u_1 - v_1) + 2m_2u_2 = m_2(u_1 + v_1)$$
$$m_1u_1 - m_1v_1 + 2m_2u_2 = m_2u_1 + m_2v_1$$
$$m_1v_1 + m_2v_1 = 2m_2u_2 + m_1u_1 - m_2u_1$$
$$v_1(m_1 + m_2) = 2m_2u_2 + (m_1 - m_2)u_1$$
$$v_1 = \frac{2m_2u_2+(m_1-m_2)u_1}{m_1+m_2}$$

$$m_1v_1 + m_2v_2 = m_1u_1 + m_2u_2$$
$$m_1v_1 = m_1u_1 + m_2u_2 - m_2v_2$$
$$m_1v_1 = m_2(u_2 - v_2) + m_1u_1$$
$$v_1 = \frac{m_2(u_2 - v_2) + m_1u_1}{m_1}$$

$$\tfrac{1}{2}m_1v_1^2 + \tfrac{1}{2}m_2v_2^2 = \tfrac{1}{2}m_1u_1^2 + \tfrac{1}{2}m_2u_2^2$$
$$m_1v_1^2 + m_2v_2^2 = m_1u_1^2 + m_2u_2^2$$
$$m_1v_1^2 = m_1u_1^2 + m_2u_2^2 - m_2v_2^2$$
$$m_1v_1^2 = m_2(u_2^2 - v_2^2) + m_1u_1^2$$
$$v_1^2 = \frac{m_2(u_2^2 - v_2^2) + m_1u_1^2}{m_1}$$

$$\left(\frac{m_2(u_2 - v_2) + m_1u_1}{m_1}\right)^2 = \frac{m_2(u_2^2 - v_2^2) + m_1u_1^2}{m_1}$$
$$\frac{m_2^2(u_2 - v_2)^2 + 2m_1m_2u_1(u_2 - v_2)^2 + m_1^2u_1^2}{m_1^2} = \frac{m_2(u_2^2 - v_2^2) + m_1u_1^2}{m_1}$$
$$m_2^2(u_2 - v_2)^2 + 2m_1m_2u_1(u_2 - v_2) + m_1^2u_1^2 = m_1m_2(u_2^2 - v_2^2) + m_1^2u_1^2$$
$$m_2^2(u_2 - v_2)^2 + 2m_1m_2u_1(u_2 - v_2) = m_1m_2(u_2^2 - v_2^2)$$
$$m_2^2(u_2 - v_2)^2 + 2m_1m_2u_1(u_2 - v_2) = m_1m_2(u_2 - v_2)(u_2 + v_2)$$
$$m_2(u_2 - v_2) + 2m_1u_1 = m_1(u_2 + v_2)$$
$$m_2u_2 - m_2v_2 + 2m_1u_1 = m_1u_2 + m_1v_2$$
$$m_2v_2 + m_1v_2 = 2m_1u_1 + m_1u_2 - m_2u_2$$
$$(m_2 + m_1)v_2 = 2m_1u_1 + (m_1 - m_2)u_2$$
$$v_2 = \frac{2m_1u_1 + (m_1 - m_2)u_2}{(+m_1}$$

2) Consider the problem of two blocks sliding on a level surface with no friction and a wall to their left, where all collisions are elastic.



Write a program in C to calculate the number of collisions. A collision is counted for block 1 hitting the wall or block 1 and 2 hitting each other. Run your program with m1 = m2 = 1, u1 = 0, u2 = 1 and verify that you get the correct answer of 3.

```c
#include <stdio.h>

double u1 = 0;
double u2 = -1;
double v1, v2;
int m1 = 1;
long long int m2 = 100000000000000;
unsigned long long int collisions = 0;

void block_collision(void);
void wall_collision(void);

int main()
{
    while (u1 > u2)
    {
        block_collision();

        if (v1 < 0)
        {
            wall_collision();
        }

        //update new initial velocities
        u1 = v1;
        u2 = v2;
    }

    printf("The number of collisions is: %llu\n", collisions);
}

void block_collision(void)
{
    v1 = (u1 * (m1 - m2) + 2 * m2 * u2) / (m1 + m2);
    v2 = (u2 * (m2 - m1) + 2 * m1 * u1) / (m1 + m2);
    collisions++;
}
```

```c
void wall_collision(void)
{
    v1 *= -1;
    collisions++;
}
```

3) Now re-run your program but successively increasing the mass of block 2 such that $m_2 \in \{100, 10000, 1000000, ...\}$. What do you discover?

```
PS G:\1201_Programming\XMAS> gcc .\PS2.c -o PS2
PS G:\1201_Programming\XMAS> .\PS2.exe
The number of collisions is: 3
PS G:\1201_Programming\XMAS> .\PS2.exe
The number of collisions is: 31
PS G:\1201_Programming\XMAS> .\PS2.exe
The number of collisions is: 314
PS G:\1201_Programming\XMAS> .\PS2.exe
The number of collisions is: 3141
PS G:\1201_Programming\XMAS> .\PS2.exe
The number of collisions is: 31415
PS G:\1201_Programming\XMAS> .\PS2.exe
The number of collisions is: 314159
PS G:\1201_Programming\XMAS> .\PS2.exe
The number of collisions is: 3141592
PS G:\1201_Programming\XMAS> .\PS2.exe
The number of collisions is: 31415926
```

I found that as I increased the mass of block 2, the ratio between the number of collisions and the mass of block 2 tended towards pi.

SCC2: Getting Started with C

Odd Numbers - 04/01/21

The aim of this program is to print out all the odd numbers from 0 to 100.

```c
#include <stdio.h>

int main()
{
    for (int i = 0; i <= 100; i++)
    {
        if ((i%2) == 1)
        {
            printf("%d\n", i);
        }
    }
}
```

```
PS G:\1201_Programming\Lancaster\Problem_Set_2> .\odd.exe
1
3
5
7
9
11
13
15
17
19
```

Squares - 01/11/20

To print a square I am using nested for loops that will print an asterisk for each row and column of the shape.

```c
#include <stdio.h>

int main()
{
    printf("What size would you like your square?\n");

    int sq_size = 0;
    scanf("%d", &sq_size); //Read size of square

    for (int i = 0; i < sq_size; i++) //print so many across and so many down
    {
        for (int j = 0; j < sq_size; j++)
        {
            printf("* ");
        }
        printf("\n");
    }
}
```

```
PS G:\1201_Programming\Lancaster\Problem_Set_2> .\square_drawer.exe
What size would you like your square?
5
*  *  *  *  *
*  *  *  *  *
*  *  *  *  *
*  *  *  *  *
*  *  *  *  *
```

Triangles - 01/11/20

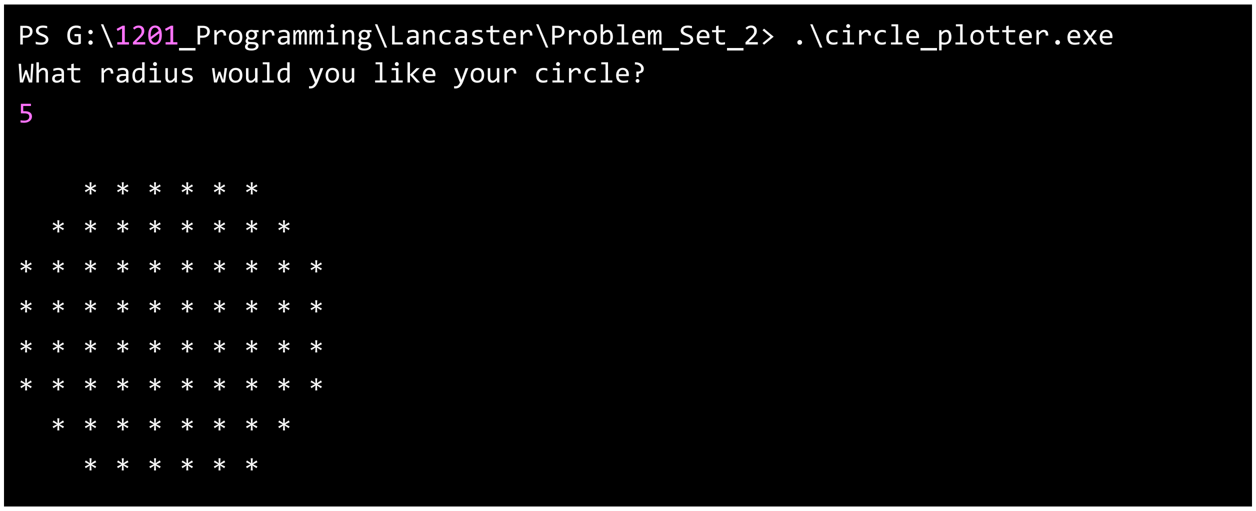
To print a triangle I make the program add a blank space to “pad out” the empty areas.

```c
#include <stdio.h>

int main()
{
    printf("What size would you like your triangle?\n");

    int tr_size = 0;
    scanf("%d", &tr_size); //Read size of square

    for (int i = 0; i < tr_size; i++) //move down a row
    {
        for (int j = 0; j < tr_size - i; j++) //print so many blank space
        {
            printf(" ");
        }
        for (int j = 0; j < (2 * i) + 1; j++) // print so many marks
        {
            printf("*");
        }
        printf("\n");
    }
}
```

```
PS G:\1201_Programming\Lancaster\Problem_Set_2> .\triangle_drawer.exe
What size would you like your triangle?
5
     *
    ***
   *****
  *******
 *********
```

More Shapes - 01/11/20

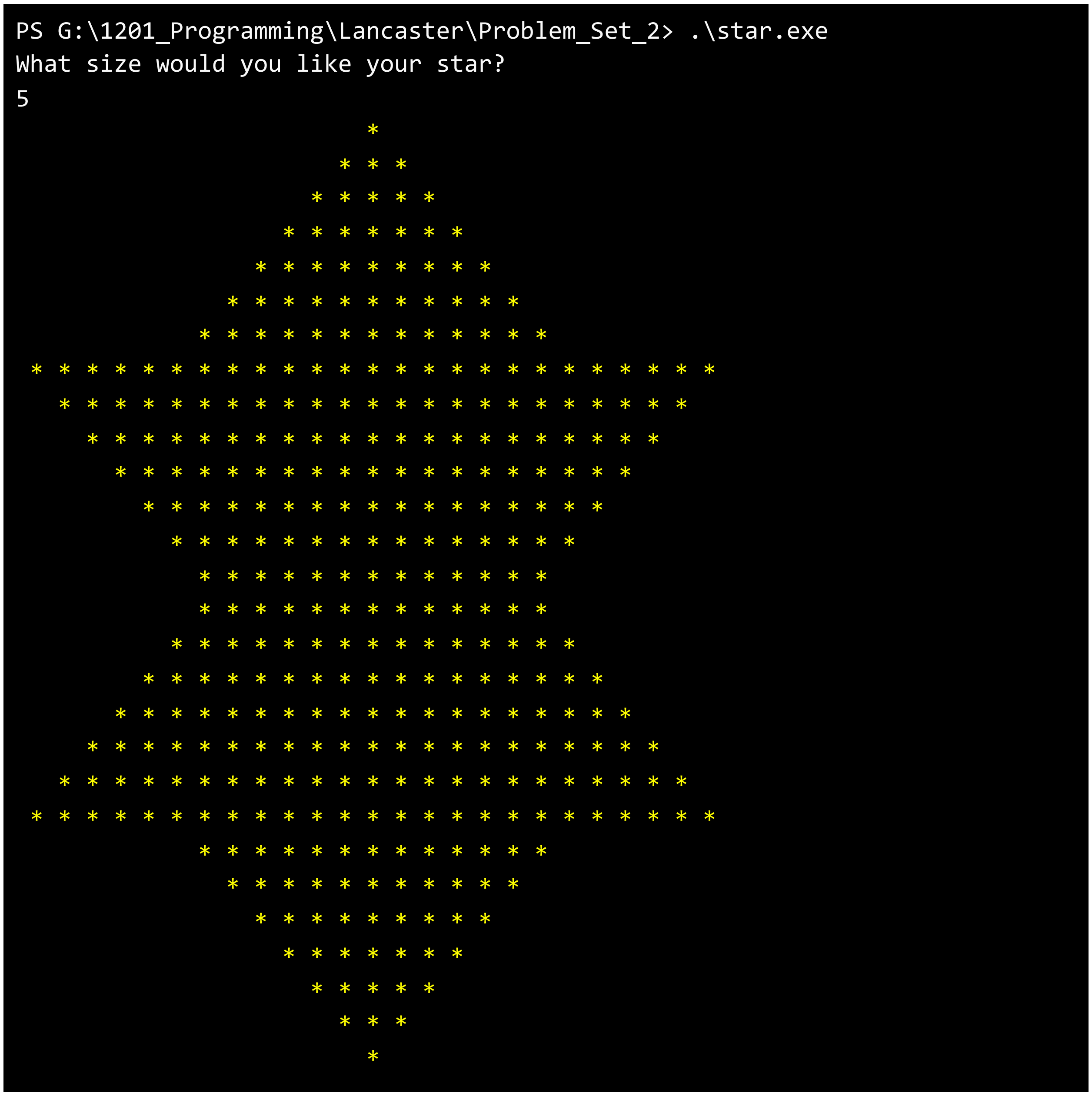
```c
#include <stdio.h>
#include <math.h>

int main()
{
    printf("What radius would you like your circle?\n");

    int rad = 0;
    scanf("%d", &rad); //Read size of square

    for (int i = 0; i <= 2 * rad; i++) //move down a row
    {
        float x = pow(pow(rad, 2) - pow(i - rad, 2), 0.5);

        for (int j = 0; j < rad - round(x); j++) //print so many blank space
        {
            printf("  ");
        }
        for (int j = 0; j < 2*round(x); j++) // print so many marks
        {
            printf("* ");
        }
        printf("\n");
    }
}
```



```
PS G:\1201_Programming\Lancaster\Problem_Set_2> .\circle_plotter.exe
What radius would you like your circle?
5

    * * * * * *
  * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
  * * * * * * * *
    * * * * * *
```

```c
#include <stdio.h>
#include <stdint.h>

void plotTri(int st_size, int spaces, int asterix);
void plotInv(int st_size, int spaces, int asterix);

int main()
{
    printf("How pointy would you like your star?\n");

    int st_size = 0;
    scanf("%d", &st_size); //Read size of square
    st_size++;

    printf("\x1B[33m"); //Colour solution from David Guyon on StackOverflow 33
= yellow

    plotTri(st_size, 2, 0);
    plotInv(st_size, 1, 2);
    plotTri(st_size, 1, 2);
    plotInv(st_size, 2, 0);

    printf("\x1B[39m"); //Colour solution from David Guyon on StackOverflow 39
= default
}

void plotTri(int st_size, int spaces, int asterix)
{
    for (int i = 0; i <= st_size; i++) //move down a row
    {
        for (int j = 0; j < spaces * st_size - i; j++) //print so many blank
space
        {
            printf("  ");
        }
        for (int j = 0; j < (2 * i) + 1 + asterix * st_size; j++) // print so
many marks
```

```c
        {
            printf(" *");
        }
        printf("\n");
    }
}

void plotInv(int st_size, int spaces, int asterix)
{
    for (int i = st_size; i >= 0; i--) //move down a row
    {
        for (int j = 0; j < (st_size - i); j++) //print so many blank space
        {
            printf("  ");
        }
        for (int j = 0; j < ((2 * i) + 2 * st_size + 1); j++) // print so many
marks
        {
            printf(" *");
        }
        printf("\n");
    }
}
```



```
PS G:\1201_Programming\Lancaster\Problem_Set_2> .\star.exe
What size would you like your star?
5
                         *
                       * * *
                     * * * * *
                   * * * * * * *
                 * * * * * * * * *
               * * * * * * * * * * *
             * * * * * * * * * * * * *
 * * * * * * * * * * * * * * * * * * * * * * * * *
   * * * * * * * * * * * * * * * * * * * * * * *
     * * * * * * * * * * * * * * * * * * * * *
       * * * * * * * * * * * * * * * * * * *
         * * * * * * * * * * * * * * * * *
           * * * * * * * * * * * * * * *
             * * * * * * * * * * * * *
             * * * * * * * * * * * * *
           * * * * * * * * * * * * * * *
         * * * * * * * * * * * * * * * * *
       * * * * * * * * * * * * * * * * * * *
     * * * * * * * * * * * * * * * * * * * * *
   * * * * * * * * * * * * * * * * * * * * * * *
 * * * * * * * * * * * * * * * * * * * * * * * * *
             * * * * * * * * * * * * *
               * * * * * * * * * * *
                 * * * * * * * * *
                   * * * * * * *
                     * * * * *
                       * * *
                         *
```

SCC3: Games Time!
https://www.cplusplus.com/reference/cstdlib/rand/

Guess The Number - 04/01/21

Gameplay is simple. If the user guesses too high, the computer tells them to guess “lower”; if the user guesses too low the computer tells them to guess “higher”. The game ends when the user guesses the number correctly.

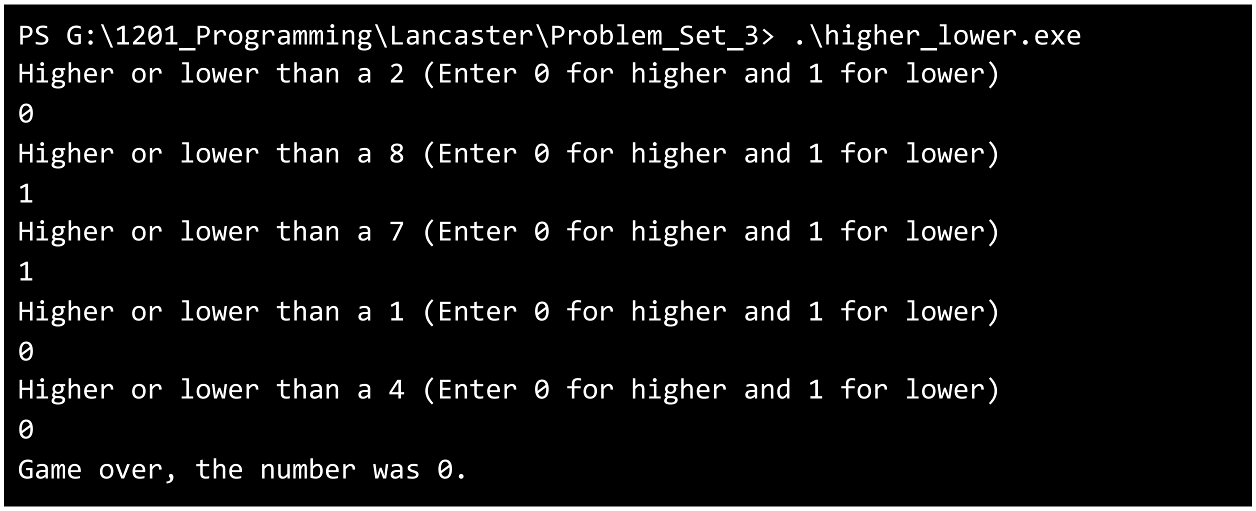
```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main()
{
    srand(time(NULL));
    int rand_num = rand() % 11; //thinks of number

    printf("I've thought of a number between 0 and 10\nEnter your guess...\n");

    for (;;)
    {
        int guess = 0;
        scanf("%d", &guess);

        if (guess < rand_num)
        {
            printf("higher\n");
        }
        if (guess > rand_num)
        {
            printf("lower\n");
        }
        if (guess == rand_num)
        {
            printf("Well done the number I thought of was %d.\n", rand_num);
            break;
        }
    }
}
```

```
PS G:\1201_Programming\Lancaster\Problem_Set_3> .\guess_number.exe
I've thought of a number between 0 and 10
Enter your guess...
5
higher
9
lower
7
higher
8
Well done the number I thought of was 8.
```

Higher or Lower

The second game is based on a similar idea to the first, but is a little more complex. For this game the idea is that the computer displays a number between 0 and 9 and then asks the user to guess if the next number the computer picks will be higher or lower.

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main()
{
    srand(time(NULL));

    int prev_num = rand() % 11;
    for (;;)
    {
        int rand_num = rand() % 11; //thinks of number
        int input = 0;

        printf("Higher or lower than a %d (Enter 0 for higher and 1 for
lower)\n", prev_num);
        scanf("%d", &input);

        if( (prev_num < rand_num) && (input == 1) ||
            (prev_num > rand_num) && (input == 0))
        {
            printf("Game over, the number was %d.", rand_num);
            break;
        }
        prev_num = rand_num;
    }
}
```



```
PS G:\1201_Programming\Lancaster\Problem_Set_3> .\higher_lower.exe
Higher or lower than a 2 (Enter 0 for higher and 1 for lower)
0
Higher or lower than a 8 (Enter 0 for higher and 1 for lower)
1
Higher or lower than a 7 (Enter 0 for higher and 1 for lower)
1
Higher or lower than a 1 (Enter 0 for higher and 1 for lower)
0
Higher or lower than a 4 (Enter 0 for higher and 1 for lower)
0
Game over, the number was 0.
```