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.

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

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.

F: > 1201_Programming > c4 > C file_reader_back001.c > main()

```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", toupper(fgetc(fp))); //reads value from file, prints value, and position along
    }
    fclose(fp);
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS F:\1201_programming\c4> gcc file_reader_back001.c -o file_reader_back001
PS F:\1201_programming\c4> .\file_reader_back001
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG.f
```

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, int alpha_array)
{
    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));
    }
}
```

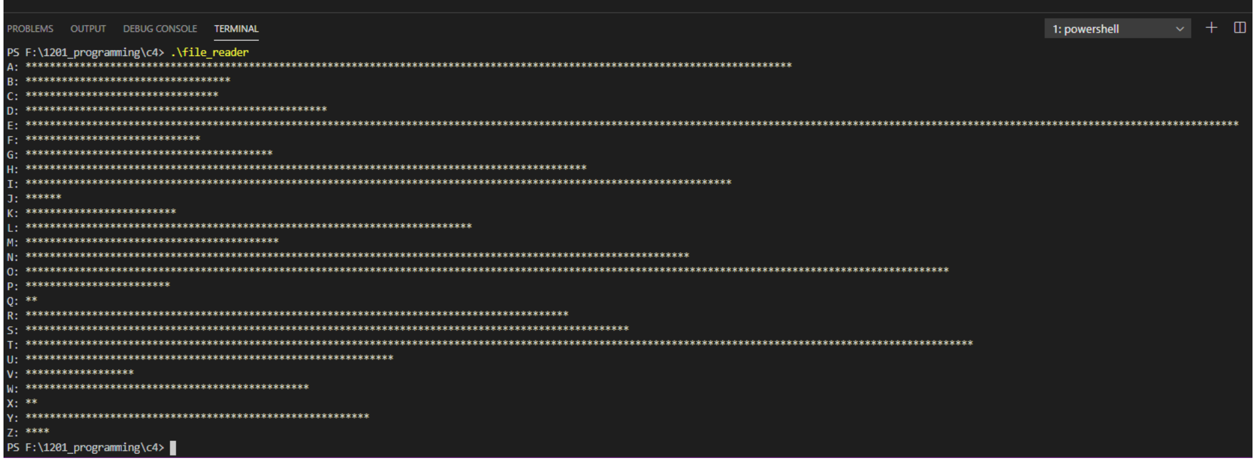
```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

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
PS F:\1201_programming\c4>
```

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");
    }
}
```



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");
}
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

JXUGKYSARHEMDVENZKCFIELUHJXUBQPOTEW

THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG.f
PS F:\1201_programming\c4> gcc c_cipher.c -o c_cipher
PS F:\1201_programming\c4> .\c_cipher
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

JXUGKYSARHEMDVENZKCFIELUHJXUBQPOTEW

THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
PS F:\1201_programming\c4>
```