**Laboratory Report**

**LABORATORY 6 & 7 – REPORT**

Student name: Michael Lenehan

Student ID: 15410402

Programme: CE

I hereby declare that the attached submission is all my own work, that it has not previously been submitted for assessment, and that I have not knowingly allowed it to be used by another student. I understand that deceiving or attempting to deceive examiners by passing off the work of another as one's own is not permitted. I also understand that using another's student’s work or knowingly allowing another student to use my work is against the University regulations and that doing so will result in loss of marks and possible disciplinary proceedings.

**Signed:** Michael Lenehan **Date:** 24 March 2016

Note: Coursework examiners are entitled to reject any coursework which does not have a signed copy of this form attached or are submitted late.

# Problem 1 (B)

The aim of this exercise was to print a series of geometric patterns using “for” loops. The programme must initially print a square, then a decreasing series of stars, then an increasing series of stars.

# Plan

Declare all necessary variables; star, line.

Establish “for” loops to print the stars, and to separate onto lines.

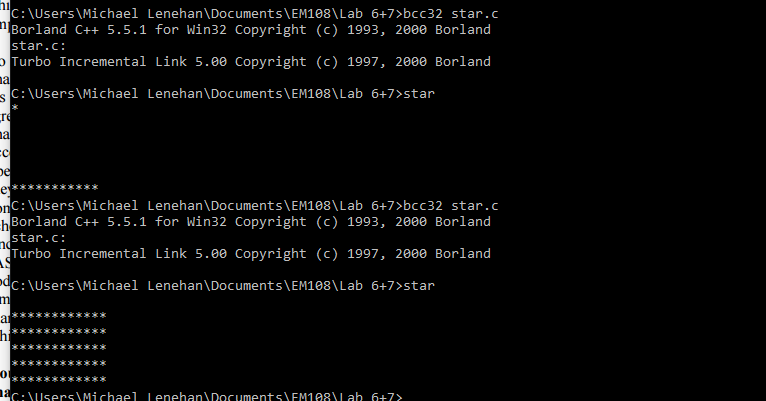
Establish “for” loops to increase and decrease amount of stars in a line.

# Development

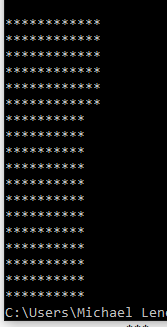
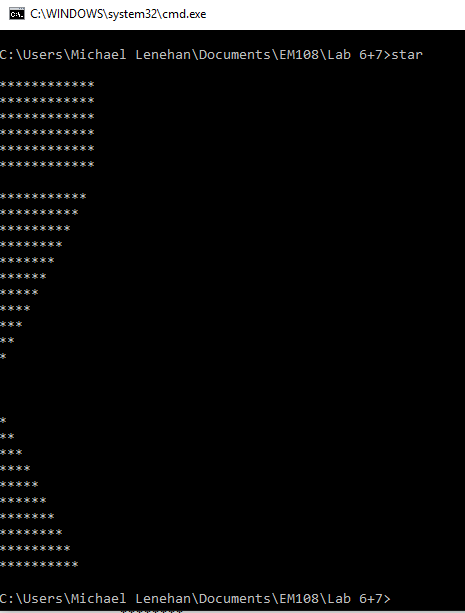
Development began with the introduction of the c libraries stdlib.h and stdio.h. The variables were then declared inside the main function. A “for” loop was then coded to decrease the number of lines necessary after each loop, and another loop was added to increase the number of stars per line to 11. This completed the first geometric pattern. For the second pattern, a “for” loop was coded to decrease the number of lines needed after each loop, and another loop was added to decrease the number of stars per line after each loop. For the third geometric pattern, the code for the second was copied and some values changed to give an increasing number of stars per line.

# Testing

Initial testing showed that the code was incorrectly looping, due to loops being in the incorrect order. This was corrected and the first geometric shape was completed.



The second geometric shape would print with equal numbers of stars per line. This was due to the incorrect implementation of a “for” loop.



Incorrect for loop was corrected, and the third geometric pattern was completed using a modified version of the second patterns code.

# Conclusion

This exercise tested the knowledge of for loops, and using multiple for loops to produce a pattern.

# Code

#include <stdlib.h>

#include <stdio.h>

int main()

{

int star, line;

for(line=5; line>=0; line--){

printf("\n");

for(star = 0; star <= 11; star ++){

printf("\*");

}

}

printf("\n\n");

for(line=11;line>=0; line --){

for(star = 0; star<line; star++){

printf("\*");

}

printf("\n");

}

printf("\n");

for(line=1;line<=11; line ++){

for(star = 1; star<line; star++){

printf("\*");

}

printf("\n");

}

return(EXIT\_SUCCESS);

}

# Problem 2

The aim of this exercise was to code a Caesar Cipher, which would encrypt a scanned in message. A user defined input was to be used to encrypt the user defined message.

# Plan

Define all necessary variables; cipher, text[20], i, key.

Provide a welcome message for the user.

Scan user input message.

Scan for user defined key.

Use a “for” loop to test for a letter value, and to add the key value to its ANSII code value, ensuring to loop back to the beginning of the alphabet.

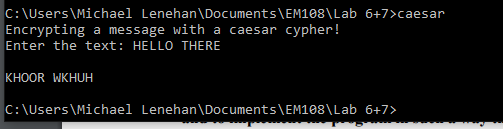
Print the encrypted message.

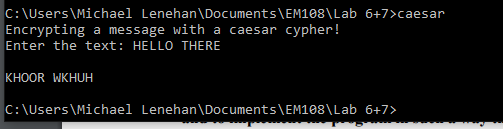
# Development

Development began with the addition of the c libraries stdio.h, stdlib.h, ctype.h, and string.h. All necessary variables were defined within the main function. The welcome message and the request for a user input message were added. The “gets()” function was used to ensure any values after a space in the string would be read. A message then requests the value for the key, which is then scanned. A “for” loop was implemented to loop through each character of the string, and an if statement added, testing for uppercase letters, and adding the key value to the ANSII code. If the character was not upper case, the no action was taken, as this value was a space. A print function was then used to print the encrypted message to the screen.

# Testing

Initial testing began using a hard coded key value of three.

The user defined input for the key was then coded, and testing was repeated to ensure that it was correctly working.

The code was deemed correctly working.

The spelling of Cipher was then corrected in the code, and a message added to advise on the amount of text which can be added, and that all text must be in uppercase.

# Conclusion

This exercise yet again tested the knowledge of for loops, but also tested for the knowledge of the gets function, the isupper function and the ability to adjust a character by changing the ANSII code.

# Code

#include<stdio.h>

#include<stdlib.h>

#include<ctype.h>

#include<string.h>

int main()

{

char cipher, text[50];

int i, key;

printf("Encrypting a message with a caesar cipher!\n");

printf("Enter the text(max 50 letters and uppercase only): ");

gets(text);

printf("\nEnter the key value: ");

scanf("%d", &key);

for(i=0; i < strlen(text); i ++)

{

if(isupper(text[i]))

{

if(text[i]+key>90){

cipher = (text[i]+key)-26;

text[i] = cipher;

}

else{

cipher = text[i]+key;

text[i] = cipher;}

}

}

printf("\n%s\n", text);

return(EXIT\_SUCCESS);

}

# Problem 3

The aim of this problem was to calculate and graph projectile motion using user defined inputs for x and y components of velocity and an initial y value. The calculated values were to be redirected to a text file, then plotted using Excel. The code was then to be broken down into shorter functions, and the file to be written to without using redirection.

# Plan

Define all necessary variables; x[100], vxo, t, y[100], yo, vyo, g.

Define values for g and t.

Print messages requesting user inputs.

Scan user inputs.

Calculate values for y and x using given variables, for increasing values of t, while y is greater than 0.

Print results.

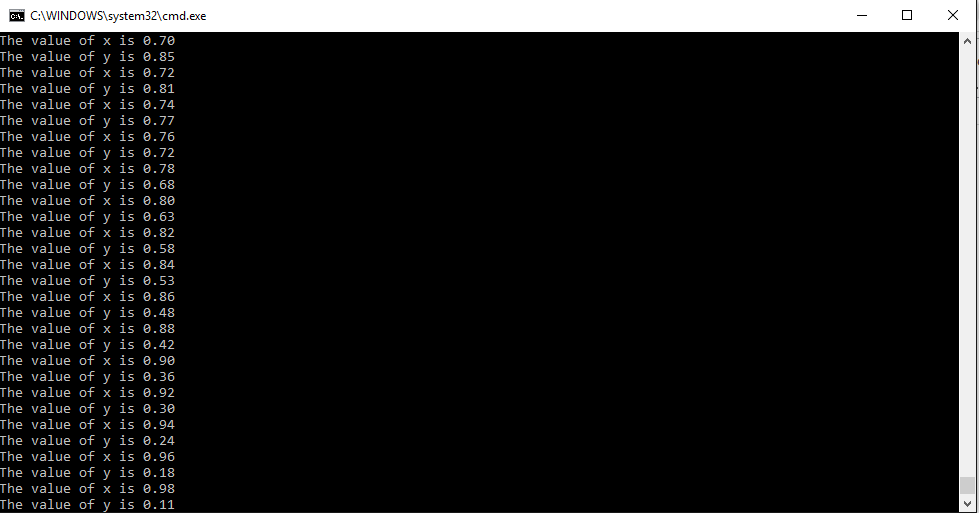
# Development

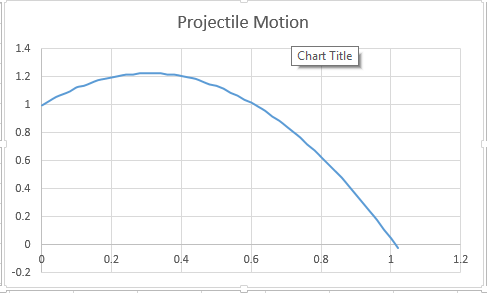
Development began with the inclusion of standard c libraries. All necessary variables were then declared. Messages were then printed requesting inputs for the variables yo, vyo, and vxo. These inputs were scanned. A while loop was then implemented to increase t while y was greater than or equal to 0. Values for y and x were printed to the screen.

# Testing

Initial testing gave continuous values of 0, due to expressions being placed outside of the while loop.

Code Corrected and Values for y and x calculated.

Graph of the projectile motion plotted.



# Conclusion

This exercise yet again tested the use of loops, in this case while loops. It also tested the use of command line redirections. The second part, which was not correctly completed on time tested the declarations of functions, and the use of pointers.

# Code

Code 1:

#include <stdio.h>

#include <stdlib.h>

int main()

{

float x[100], vxo, t, y[100], yo, vyo, g;

g = 9.81;

yo = 0;

t = 0;

printf("Enter vertical height: \n");

scanf("%f", &yo);

printf("Enter horizontal velocity component: \n");

scanf("%f", &vxo);

printf("Enter vertical velocity component: \n");

scanf("%f", &vyo);

while(y[t]>=0){

x[t] = (vxo\*t);

y[t] = yo +vyo\*t -(g\*t\*t);

printf("%1.2f\t%1.2f\n", x[t],y[t]);

t=t+0.01;

}

return(EXIT\_SUCCESS);

}