# Week 3 Assessment

**…**

**Code Task:** Please paste the code below **(10 marks)**:

#include <iostream>

using namespace std;

#include <iostream>

using namespace std;

int main() {

int height; char outputChar;

cout << "enter single digit: "; cin >> height;

while (height < 1 || height > 9) { // force height to be in the legal range

if (height == 0) {

// special case for null pyramids because 0 is technically a single digit

cerr << "error: pyramid of height 0 is impossible" << endl;

// this error message used to be a lot saltier before i decided it was best to stay

// relatively professional

}

else {

cerr << "error: invalid input" << endl;

}

cout << "enter single digit: "; cin >> height;

}

cout << endl;

for (int8\_t i = 1; i <= height; i++) {

// int8's used because, let's face it, we don't need 32 bits per index

for (int8\_t j = 1; j < 2 \* height; j++) {

if (j <= height - i || j >= height + i) {

outputChar = ' '; // if (i, j) is outside the pyramid, output a space

}

else {

// otherwise, find outputChar mathematically

if (j <= height) {

outputChar = i - height + j;

}

else {

outputChar = height + i - j;

}

outputChar += 0x30; // add ord('0') to outputChar to turn it into a printable number

}

cout << outputChar;

}

cout << endl;

}

return 0;

}

**Report Task:** …

First, the program takes in input from the user and stores it in height. It also declares a variable called outputChar for use later. While height is outside the legal range, it spits out an error and takes input again. Then, it generates the pyramid based on the value of height. This is done with two loops – one inside the other. Inside the inner loop, if j is not in between height - i and height + i, it sets outputChar to a space, otherwise it checks whether j is less than or equal to height. If it is, it sets outputChar to i - height + j (see the table for where that came from), otherwise it sets it to height + i - j.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Let n = height = 2. Then… | | | | |
|  | Output for j=... | | |  |
| i | 1 | 2 | 3 | i-n |
| 1 | 0 | 1 | 0 | -1 |
| 2 | 1 | 2 | 1 | 0 |

Finally, the else block adds 0x30, the ASCII code for zero, to outputChar to turn it into a printable number before outputting it. After the inner loop completes, it prints a line break before the next iteration of the outer loop. After the outer loop completes, the program terminates. The output is shown below.

A computer screen with white text

AI-generated content may be incorrect.

The original idea involved three inner loops. This was scrapped before any evidence of this step was collected on the grounds that it was among the worse ideas the author has had in their entire life. Two of the inner loops were removed, leaving just the one. Both loops were from one to height, but the bounds of the inner loop were quickly corrected to being from one to double the value of height. outputChar was originally set to j-i if j was in range, purely as a test value to be corrected later, giving the output shown below.

A black screen with white text

AI-generated content may be incorrect.

This is when the author realised that a single if/else block was insufficient, leading to the if/else nested inside the else statement in the final program. outputChar was temporarily set to 0 if j was greater than height at this stage, to get the left side working first before worrying about the right. The correct value of outputChar for j > height was readily apparent once it was correct for j <= height.