Vectors

Multiply Matrices constructor

```
std::vector<std::vector<int>> multiplyMatrices
(std::vector<std::vector<int>>& mat1,std::vector<std::vector<int>>& mat2){
```

Product constructor

```
std::vector<std::vector<int>> product(mat1.size(),
 std::vector<int>(mat2[0].size(), 0));
```

`std::vector<int>(mat2[0].size(), 0));`
is the second argument to the outer vector's constructor,
which specifies what each element of the outer vector should be.
 In this case, each element is itself a vector.

`mat2[0].size()`
means "get the size of the first row of mat2" which gives us the number of columns in mat2.
 This is how many columns our result matrix needs.
The `0` at the end is the value that will fill every position in each inner vector.
 initializes all elements of the matrix to zero.

Dot product loop

```
for (size_t i = 0; i < mat1.size(); i++) {
    // Iterate through each column with   only size of first row of mat2

    for (size_t j = 0; j < mat2[0].size(); j++) {
        // Iterate through each element of the row of mat1 and column of mat2
        for (size_t k = 0; k < mat2.size(); k++) {
            product[i][j] += mat1[i][k] * mat2[k][j];
        }
    }
}
return product;
}
```

Output methods

Return string

```
std::string generateRightJustifiedTriangle(char c, int N) {
    std::string answer;
    int j = 1;

    for (int i = N; i > 0; i--) {
        // Use std::string(count, character) constructor correctly
        answer += std::string(i - 1, ' ') + std::string(j, c);
        answer += "\n";  // Add newline for each row
        j++;
    }
    return answer;
}
```

`std::string(i - 1, ' ')` creates a new string containing `i - 1` space characters.
For example, if `i` is 5, this creates a string with 4 spaces: " "
This forms the left side of each row (the spaces)
`std::string(j, c)` creates a new string containing `j` copies of the character `c`.
For example, if `j` is 1 and `c` is '', *this creates: ""*
This forms the right side of each row (the triangle characters)

Stringstream

```
 #include <iostream>
#include <string>
#include <iomanip>
#include <sstream>
std::string generateRightJustifiedTriangle(char c, int N) {
 std::stringstream ss; for (int i = 1; i <= N; i++) {
// Set the width to N and right-justify
ss << std::setw(N) << std::right; // Create a string of i char
std::string row(i, c);
// Add the row to the stringstream ss << row; // Add a newline unless it's
the last row
if (i < N) {
ss << '\n';
 }
} return ss.str();
}
```

C-string methods

`strcmp()`

```
char s1[] = "apple";
char s2[] = "orange";
int result = strcmp(s1, s2); // Returns negative value
```

`strcpy()`

```
char source[] = "Hello";   char destination[10];
strcpy(destination, source); // destination now contains "Hello"
```

`strlen()`

```
char name[] = "Hello";    int length = strlen(name); // Returns 5
```

Well, the wanted