



Question 1

(i)

```
/**
 * Check function header
 * @param n1 - integer number
 * @param n2 - float number
 * @pre - N/A function was not defined in exercise
 * @post - N/A function was not defined in exercise
 */
void check(int n1, float n2);
```

(ii)

```
/**
 * Multiplies 2 floats and returns the product
 * @param n1 - The first float.
 * @param n2 - The Second float.
 * @pre - Parameters are not undefined before function call.
 * @return - The product of n1 times n2.
 */
float mult(float n1, float n2);
```

(iii)

```
#include <string>
/**
 * @brief Time a recursive function
 * @param s - Seconds
 * @param m - Minutes
 * @param h - Hours
 * @pre - Parameters are not undefined before function call.
 * @return - A string representation of the time.
 * @note - Return type was not defined in exercise, I assume function returns string.
 */
std::string time(int s, int m, int h);
```

(iv)

```
#include <string>
/**
 * @brief Counts the number occurrences of a character in a string.
 * @param s - String parameter to be searched.
 * @param c - Search key character.
 * @pre - Actual parameters s and c are defined before function call.
 * @return - int count of occurrence of 'c' in 's'.
 */
int countChar(std::string s, char c);
```

Question 2

(i)

/*

Solution: move function2 definition outside of function1 definition.
The proper using directives and include directives are not present.
Code should look like this:

```
#include <iostream>
```

```
using namespace std;
```

```
// Function 2 definition outside of function 1
```

```
void function2() {  
    cout << "Inside function function1 " << endl;  
};
```

```
// Function 1 definition without function 2 being defined in it
```

```
void function1() {  
    cout << "Inside function function1 " << endl;  
    // Call function 2 if you want. Will be available in global scope  
    function2();  
};  
*/
```

```
#include <iostream>
```

```
using namespace std;
```

```
// Function 2 definition outside of function 1
```

```
void function2() {  
    cout << "Inside function function1 " << endl;  
};
```

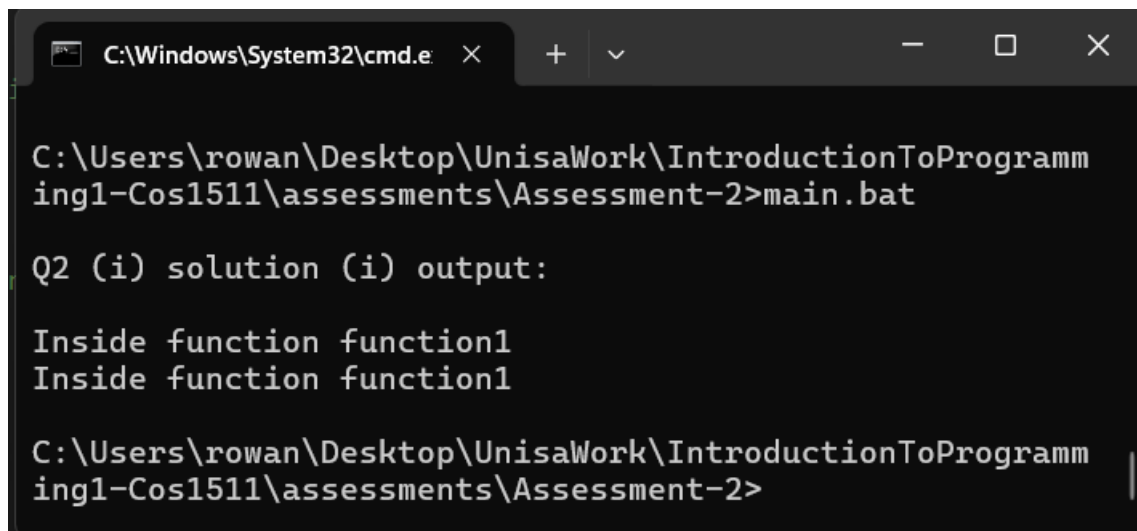
```
// Function 1 definition without function 2 being defined in it
```

```
void function1() {  
    cout << "Inside function function1 " << endl;  
    // Call function 2 if you want. Will be available in global scope  
    function2();  
};
```

```
// Test solution
```

```
int main() {  
    // Format console output  
    cout  
        << endl  
        << "Q2 (i) solution (i) output:" << endl  
        << endl  
    ;  
    // Calling function1 for testing purposes  
    function1();  
    return 0;  
};
```

Output:



```
C:\Windows\System32\cmd.e  X  +  v  -  □  X

C:\Users\rowan\Desktop\UnisaWork\IntroductionToProgramming1-Cos1511\assessments\Assessment-2>main.bat

Q2 (i) solution (i) output:

Inside function function1
Inside function function1

C:\Users\rowan\Desktop\UnisaWork\IntroductionToProgramming1-Cos1511\assessments\Assessment-2>
```

(ii)

/*

Error:

This is a syntax error where the sum function definition did not return an integer.
The function header required an integer to be returned.
Attempted compilation will result in a compiler error.

Solution:

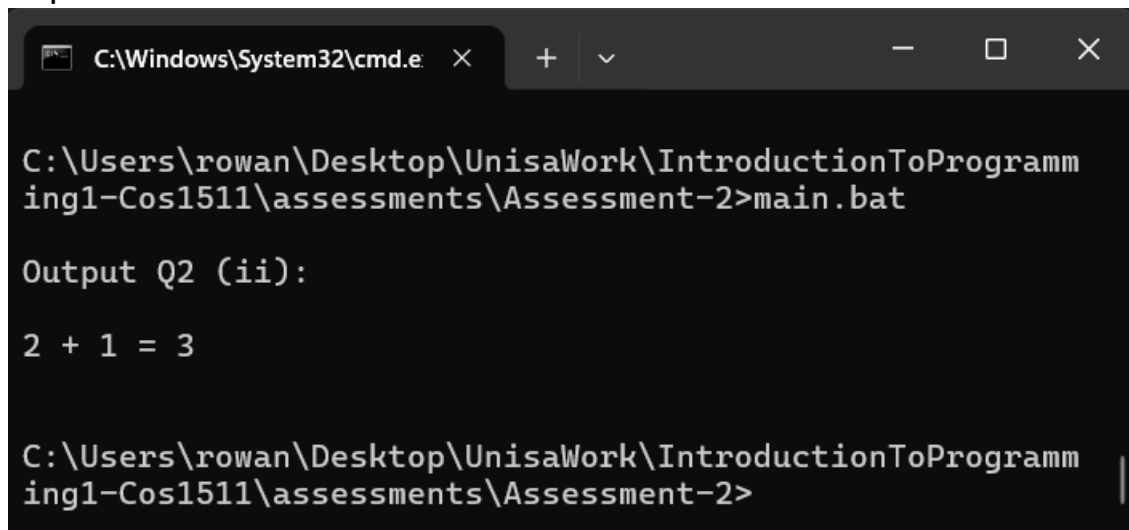
Make the function return an integer representing the sum of the 2 parameters.
Remove the result variable because it is redundant code based on its current use.
You could also add const keywords for the parameters to indicate that the function is not supposed to mutate the parameters.

The function should look like this:

```
int sum(int x, int y) {
    return x + y;
};
*/
#include <iostream>
/**
 * Adds 2 integers together and returns the sum
 * @param x - The first integer
 * @param y - The second integer
 * @pre - Actual parameters are not undefined after function call.
 * @return - The integer sum of x + y
 */
int sum(int x, int y) {
    return x + y;
};
// Adds 2 integers together
int main() {
    using namespace std;

    // Define test variables
    const int
        TEST_NUMBER_1 = 2,
        TEST_NUMBER_2 = 1
    ;
    // Test console output
    cout
        << endl
        << "Output Q2 (ii):" << endl
        << endl
        << TEST_NUMBER_1 << " + " << TEST_NUMBER_2 << " = "
        << sum(TEST_NUMBER_1, TEST_NUMBER_2) << endl
        << endl
    ;
    return 0;
};
```

Output:



```
C:\Windows\System32\cmd.e  X  +  v  -  □  X

C:\Users\rowan\Desktop\UnisaWork\IntroductionToProgramming1-Cos1511\assessments\Assessment-2>main.bat

Output Q2 (ii):

2 + 1 = 3

C:\Users\rowan\Desktop\UnisaWork\IntroductionToProgramming1-Cos1511\assessments\Assessment-2>
```

(iii)

/*

Error:

This is a syntax error and a logic error

It is very hard to say what the error is since the code is written so poorly.

The function seems to be an attempt to write a recursive factorial function for integers even though it's called computeProd.

I don't know if this is allowed in answering the question but I would also rename the function to intFactorial or intFact.

The new heading would be more descriptive for the purpose of the function vs computeProd.

Syntax error:

```
1 int computeProd(int n) {
2     if (n == 0)
3         return 0;
4     else
5     }
```

The } on line 5 is placed too early and should be placed after line 6.

There should also be a ; after the closing } for the function definition (Not strictly required but I prefer to do it).

Attempted compilation will result in a compilation error.

The code should look like this:

```
1 int computeProd(int n) {
2     if (n == 0)
3         return 0;
4     else
5         return n * computeProd(n - 1);
6     };
```

Logic error:

The function will always output 0 because the base case is when n == 0.

Therefore the last stack frame will return 0.

Therefore the 2nd last stack frame will return n * 0; which will always be 0.

This will result in each subsequent stack frame returning 0.

Then the first stack frame will return 0.

An easy fix would be to return n once n == 1 is reached, therefore changing the base case.

The code should look like this:

```
1 int computeProd(int n) {
2     if (n == 0)
3         return 0;
4     else if (n == 1) {
5         return n;
6     }
7     return n * computeProd(n - 1);
8 }
```

*/

#include <iostream>

/**

* Computes the factorial of an integer

* @param n - The integer to find the factorial of

* @pre - Actual parameter n should be assigned a value before function call (not undefined)

* @return - The factorial of the number

*/

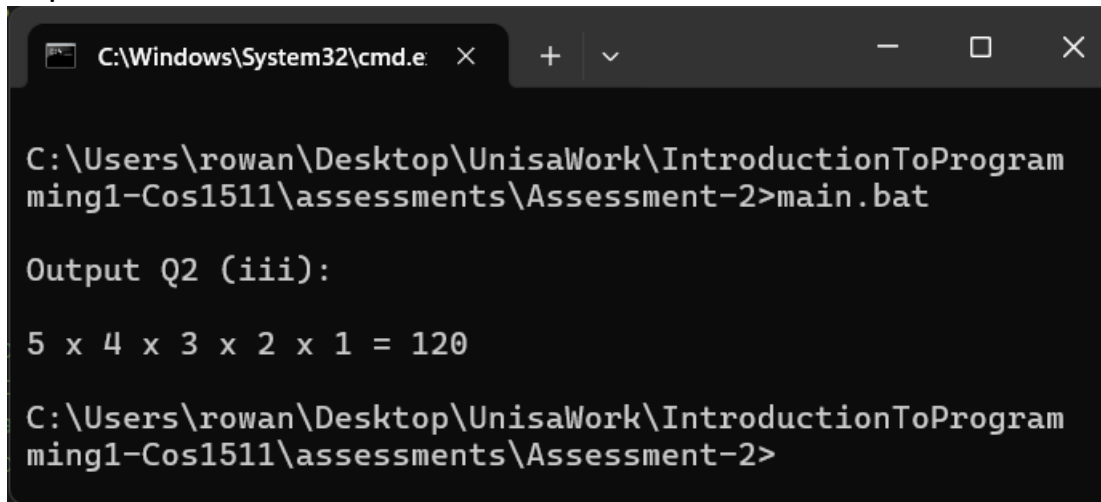
```
int computeProd(int n) {
    if (n == 0)
        return 0;
    else if (n == 1) {
        return n;
    } else
        return n * computeProd(n - 1);
};
```

// Tests solution

```
int main() {
```

```
using namespace std;
const int
    TEST_NUMBER = 5
;
// Test console output
cout
    << endl
    << "Output Q2 (iii):" << endl
    << endl
;
// Outputs 6 x 5 x 4 ... x 1 to console
cout << TEST_NUMBER;
for (int i = TEST_NUMBER - 1; i >= 1; i--) {
    cout << " x " << i;
};
// Tests final console output
cout << " = " << computeProd(5);
return 0;
};
```

Output:



The screenshot shows a Windows command prompt window with the title bar "C:\Windows\System32\cmd.e" and standard window controls. The command prompt is at the directory "C:\Users\rowan\Desktop\UnisaWork\IntroductionToProgramming1-Cos1511\assessments\Assessment-2". The user has entered the command "main.bat". The output of the program is displayed as follows:

```
C:\Users\rowan\Desktop\UnisaWork\IntroductionToProgram
ming1-Cos1511\assessments\Assessment-2>main.bat

Output Q2 (iii):

5 x 4 x 3 x 2 x 1 = 120

C:\Users\rowan\Desktop\UnisaWork\IntroductionToProgram
ming1-Cos1511\assessments\Assessment-2>
```

(iv)

/*

Error:

There is a logic error in the code.

The parameter 'a' is shadowed by the local variable 'float a' within the function definition. This will result in variable 'a' being an undefined value because it is not assigned a value after that.

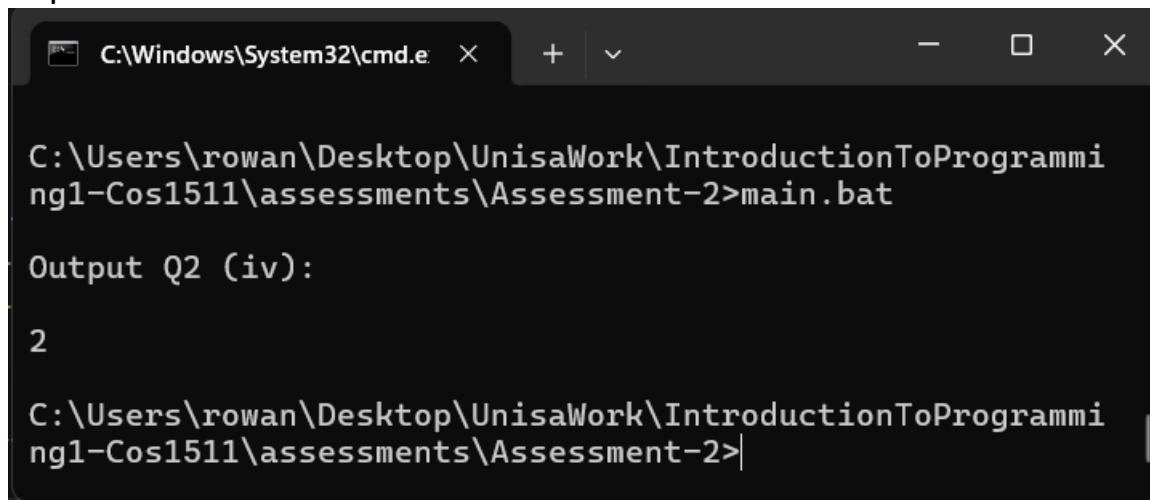
The console output should be a random float value every function call (undefined behaviour).

The proper using directives and include directives are not present.

Code should look like this:

```
#include <iostream>
void aFunction(float a) {
    using namespace std;
    cout << a << endl;
};
*/
#include <iostream>
/**
 * Outputs float to console
 * @param a - float parameter
 * @post - outputs float a to console
 */
void aFunction(float a) {
    using namespace std;
    cout << a << endl;
};
// Tests solution
int main() {
    using namespace std;
    const float
        TEST_NUMBER = 2.f
    ;
    // Format console
    cout
        << endl
        << "Output Q2 (iv):" << endl
        << endl
    ;
    // Test console output
    aFunction(TEST_NUMBER);
    return 0;
};
```

Output:



```
C:\Windows\System32\cmd.e
C:\Users\rowan\Desktop\UnisaWork\IntroductionToProgramming1-Cos1511\assessments\Assessment-2>main.bat
Output Q2 (iv):
2
C:\Users\rowan\Desktop\UnisaWork\IntroductionToProgramming1-Cos1511\assessments\Assessment-2>
```

(v)

/*

Error:

There are 2 syntax errors

Syntax errors:

1.

The function header defines theProduct as void return type meaning it does not return anything.

In the function definition it is trying to return an integer.

This is not allowed in C++.

Attempted compilation will result in a compilation error.

2.

The proper using and include directives are not used

Solution:

Remove the 'return result;' statement.

Add #include <iostream> before the function definition.

Add 'using namespace std;' either in the global scope or within the function definition.

Code should look like this:

Note: I rewrote the functions variable declarations to improve readability by removing redundant statements.

```
#include <iostream>
```

```
// alternatives is to place the using directive here
```

```
// using namespace std;
```

```
void theProduct() {
    using namespace std;
    int
        a,
        b,
        c,
        result
    ;
    cout << "Enter three integers " << endl;
    cin >> a >> b >> c;
    result = a * b * c;
    cout << "Result is " << result << endl;
};
*/
```

```
#include <iostream>
```

```
/**
```

```
 * @brief Prompts user to input 3 integers and outputs the result to the console
```

```
 * @post Outputs product of 3 input integers to the console
```

```
 */
```

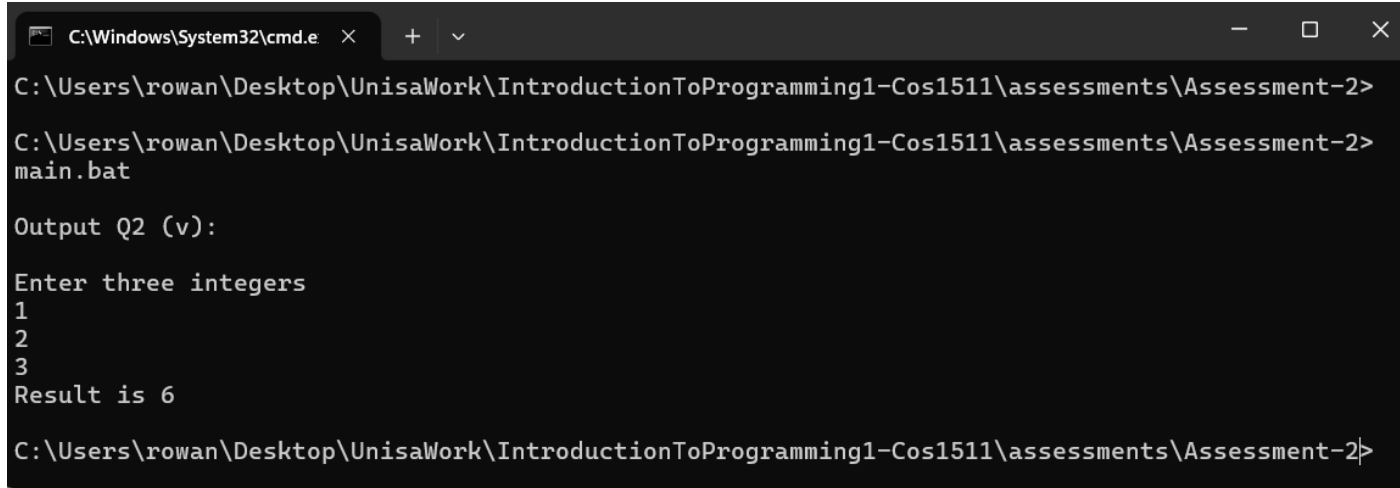
```
void theProduct() {
    using namespace std;
    int
        a,
        b,
        c,
        result
    ;
    cout << "Enter three integers " << endl;
    cin >> a >> b >> c;
    result = a * b * c;
    cout << "Result is " << result << endl;
};
```

```
// Test solution
```

```
int main() {
    using namespace std;
```

```
// Format console
cout
    << endl
    << "Output Q2 (v):" << endl
    << endl
;
// Test function call
theProduct();
return 0;
};
```

Output:



The screenshot shows a Windows Command Prompt window with the title bar 'C:\Windows\System32\cmd.e'. The window contains the following text:

```
C:\Users\rowan\Desktop\UnisaWork\IntroductionToProgramming1-Cos1511\assessments\Assessment-2>
C:\Users\rowan\Desktop\UnisaWork\IntroductionToProgramming1-Cos1511\assessments\Assessment-2>
main.bat

Output Q2 (v):

Enter three integers
1
2
3
Result is 6

C:\Users\rowan\Desktop\UnisaWork\IntroductionToProgramming1-Cos1511\assessments\Assessment-2>
```


(vi)

/*

Error:

There are no errors.

There are some code improvements that could be implemented.

The function allows invalid parameter use during runtime.

The function has no documentation.

Solution:

Add #include and #include<cmath> directives.

Add Runtime error checking for invalid parameters

Code should look like this:

```
#include <iostream>
```

```
#include <cmath>
```

```
float calculateSquare(const float number) {
```

```
    using namespace std;
```

```
    // Handles run time error checking
```

```
    if (isnan(number) || isinf(number)) {
```

```
        cout << "Warning: number is nan (undefined number) or infinity, returning nan";
```

```
        return NAN;
```

```
    };
```

```
    // Calculates square
```

```
    return number * number;
```

```
};
```

```
*/
```

```
#include <iostream>
```

```
#include <cmath>
```

```
/**
```

```
 * @brief Calculates the square of a given float number.
```

```
 * @param number The input value to be squared (must be a finite number).
```

```
 * @return The square of the input number.
```

```
 * @note Returns NaN if input is NaN or infinity.
```

```
 */
```

```
float calculateSquare(const float number) {
```

```
    using namespace std;
```

```
    // Handles run time error checking
```

```
    if (isnan(number) || isinf(number)) {
```

```
        cout << "Warning: number is nan (undefined number) or infinity, returning nan";
```

```
        return NAN;
```

```
    };
```

```
    // Calculates square
```

```
    return number * number;
```

```
};
```

```
int main() {
```

```
    using namespace std;
```

```
    // Define test variable
```

```
    const float
```

```
        TEST_NUMBER = 2.f
```

```
;
```

```
    // Format console
```

```
    cout
```

```
        << endl
```

```
        << "Output Q2 (vi):" << endl
```

```
        << endl
```

```
;
```

```
    cout.setf(ios::fixed);
```

```
    // Test console output
```

```
    cout
```

```
        << "(" << TEST_NUMBER << ")^2 = " << TEST_NUMBER << " x " << TEST_NUMBER << " = " <<
```

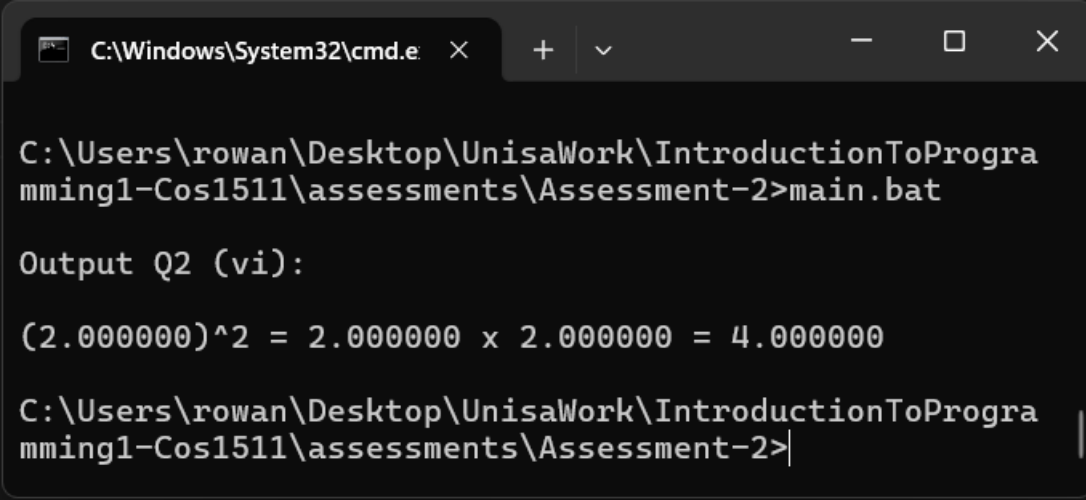
```
    calculateSquare(TEST_NUMBER) << endl
```

```
;
```

```
    return 0;
```

```
};
```

Output:



```
C:\Windows\System32\cmd.e  X  +  v  -  □  X

C:\Users\rowan\Desktop\UnisaWork\IntroductionToProgramming1-Cos1511\assessments\Assessment-2>main.bat

Output Q2 (vi):

(2.000000)^2 = 2.000000 x 2.000000 = 4.000000

C:\Users\rowan\Desktop\UnisaWork\IntroductionToProgramming1-Cos1511\assessments\Assessment-2>|
```

Question 3

(i)

```
#include <iostream>
/**
 * Cubes an integer
 * @param n - Integer number to be cubed
 * @pre - Actual parameter n is defined and assigned a value before function call
 * @return the cube of the integer n
 */
int intCube(int n) {
    return n * n * n;
};
```

(ii)

```
/**
 * Calculates the sum and difference between 2 integers and assigns the corresponding values
 * to reference parameters
 * @param n1 - First integer
 * @param n2 - Second integer
 * @param sum [in,out] Reference to the sum to be updated.
 * @param diff [in,out] Reference to the diff to be updated.
 */
void calcSumAndDiff(const int n1, const int n2, int& sum, int& diff) {
    sum = n1 + n2;
    if (n1 <= n2) {
        diff = n1 - n2;
    } else {
        diff = n2 - n1;
    };
};
```

(iii)

```
#include <iostream>
#include <string>

using std::string;

/**
 * Outputs a row to the console with parameterised filler and border.
 * Helper function
 * @param l - integer length of row.
 * @param filler - string of characters printed between the start and end of a row l times.
 * @param b - stands for border. string of characters printed at the start and end of a row.
 * @post - Outputs a row to the console with parameterised filler and border.abort
 */
void printRow(int l, const string filler = "*", const string b = "") {

    using namespace std;

    // Runtime error handling for parameter l
    if (l <= 0) {
        cout << "Warning: l must be greater than or equal to 1, assigning l = 1" << endl;
        l = 1;
    };

    // If l is 1 then only 1 character should be output to console
    if (l == 1) {
        cout << b << endl;
        return;
    };

    // Outputs border characters at the start of row to console
    cout << b;

    // Outputs filler between start and end of row to console
    for (int i = 0; i < l - 2; i++) {
        cout << filler;
    };

    // Outputs border characters at the end of row to console
    cout << b << endl;

};

/**
 * @brief - Outputs an empty rectangle with the border made of * to the console
 * @param l - integer length of the rectangle
 * @param h - integer height of the rectangle
 * @post - Outputs an empty rectangle with the border made of * to the console
 */
void rectangle(int w, int h) {

    using namespace std;

    // Runtime error handling for parameter h
    if (h <= 0) {
        cout << "Warning: h must be greater than or equal to 1, assigning h = 1" << endl;
        h = 1;
    };

    // If h is 1 then only 1 row should be output to console
    if (h == 1) {
        printRow(w);
        return;
    };
};
```

```

// Outputs top row to console
printRow(w);

// Outputs rows in between top and bottom to console
for (int j = 0; j < h - 2; j++) {
    printRow(w, " ");
};

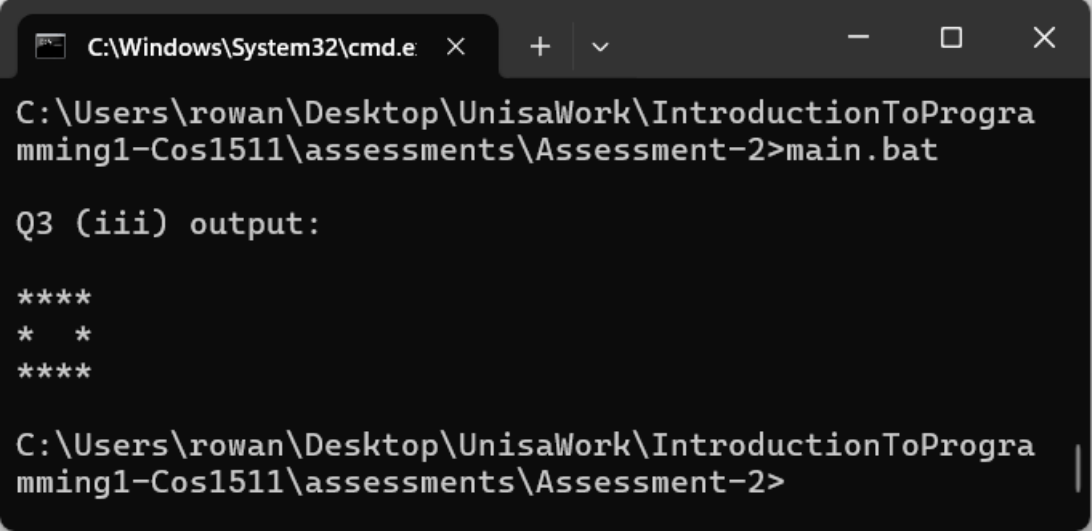
// Outputs bottom row to console
printRow(w);

};

// Test solution
int main() {
    using namespace std;
    // Format console
    cout
        << endl
        << "Q3 (iii) output:" << endl
        << endl
    ;
    // Test rectangle
    rectangle(4, 3);
    return 0;
};

```

Output:



```

C:\Windows\System32\cmd.e
C:\Users\rowan\Desktop\UnisaWork\IntroductionToProgramming1-Cos1511\assessments\Assessment-2>main.bat

Q3 (iii) output:

****
*  *
****

C:\Users\rowan\Desktop\UnisaWork\IntroductionToProgramming1-Cos1511\assessments\Assessment-2>

```

(iv)

```
#include <iostream>
/**
 * Calculates the price of a pizza
 * @param pizzaSize - Char S, M or L indicating the correponding pizza size
 * @param numToppings - integer (can only be postive or 0) indicating the number of pizza
topping selections on the pizza
 * @return
 * @note pizzaSize should be a char enum type to limit the possible options of what can be
**entered but the exercise indicated to use
 * a char so I am doing only what I am told
 */
float computePrice(const char pizzaSize, const int numToppings = 0) {

    if (numToppings < 0) {
        std::cout
            << "Warning: numToppings is negative, returning 0.f" << std::endl;
        return 0.f;
    };

    float
        price = 0.f
    ;

    // Calculates price of pizza based on pizzaSize and numToppings
    switch (pizzaSize) {
        case 'S':
            price = 50.f + (5.50f * numToppings);
            break;
        case 'M':
            price = 70.f + (6.50f * numToppings);
            break;
        case 'L':
            price = 90.f + (7.50f * numToppings);
            break;
        default:
            std::cout << "Warning: pizza size is invalid returning 0";
            return 0.f;
    };

    return price;

};
```

Question 4:

```
#include <iostream>
#include <string>
#include <cctype>

// Brings std::string into the global namespace
using std::string;

/**
 * @brief Collects address information from user input and stores it in provided references.
 *
 * This function prompts the user to enter their name/title, address lines, and postal code.
 * Each input is stored in the corresponding reference parameter. The function uses
 * getline() to allow for spaces in the input fields.
 *
 * @param[out] n      Reference to store the name/title input
 * @param[out] adr1   Reference to store the first address line input
 * @param[out] adr2   Reference to store the second address line input (can be empty)
 * @param[out] pC     Reference to store the postal code input
 *
 * @note Input is collected using getline(), so entries can contain spaces.
 * @note The function performs no validation on the input format.
 *
 * @post All reference parameters will contain the user's input values.
 * @post The input buffer will be empty after each getline() call.
 */
void inputData(string& n, string& adr1, string& adr2, string& pC) {

    using namespace std;

    // Prompts user to enter title
    cout << "Enter title: ";
    getline(cin, n);

    // Prompts user to enter address 1
    cout << "Enter address 1: ";
    getline(cin, adr1);

    // Prompts user to enter address 2
    cout << "Enter address 2: ";
    getline(cin, adr2);

    // Prompts user to enter address 1
    cout << "Enter postal code: ";
    getline(cin, pC);

};

/**
 * @brief Displays address data to console
 * @param n - name string, used in output
 * @param a1 - address 1, used in output
 * @param a2 - address 2, used in output
 * @param pC - postal code, used in output
 * @post Displays address data to console
 */
void displayData(const string n, const string a1, const string a2, const string pC) {

    using namespace std;

    // Displays data in format
    cout
        << n << endl
        << a1 << endl
        << a2 << endl
```

```

        << pC << endl
    ;

};

// Test solution
int main() {

    using namespace std;

    // Define variables
    string
        name = "",
        addr1 = "",
        addr2 = "",
        postalCode = ""
    ;

    // Format console
    cout
        << endl
        << "Q4 output:" << endl
        << endl
    ;

    // Prompts user to enter address data
    inputData(name, addr1, addr2, postalCode);

    // Format console
    cout << endl;

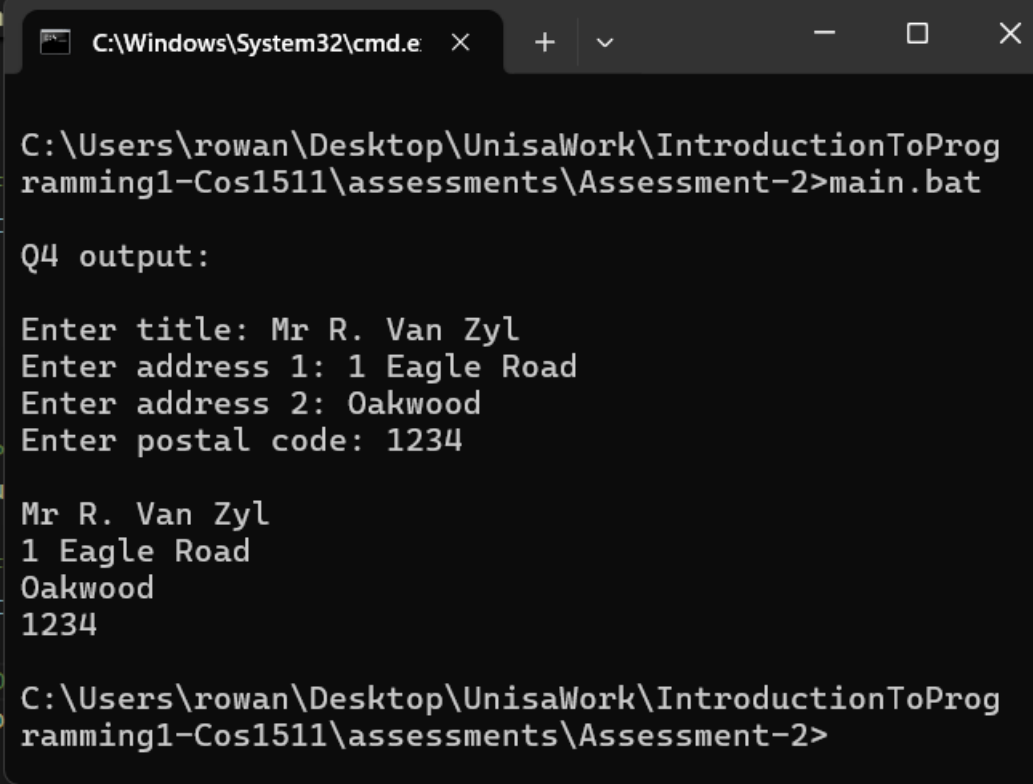
    // Outputs data to console
    displayData(name, addr1, addr2, postalCode);

    return 0;

};

```

Output:



```

C:\Windows\System32\cmd.e
C:\Users\rowan\Desktop\UnisaWork\IntroductionToProgramming1-Cos1511\assessments\Assessment-2>main.bat
Q4 output:
Enter title: Mr R. Van Zyl
Enter address 1: 1 Eagle Road
Enter address 2: Oakwood
Enter postal code: 1234

Mr R. Van Zyl
1 Eagle Road
Oakwood
1234

C:\Users\rowan\Desktop\UnisaWork\IntroductionToProgramming1-Cos1511\assessments\Assessment-2>

```

Question 5

```
#include <iostream>
#include <iomanip>

/**
 * Prompts user to input score between 0 and 100 and reprompts user if input is invalid
 * @return - integer score
 */
int getScore() {

    using namespace std;

    int s = 0;

    // Prompts user to input score between 0 and 100 and reprompts user if input is invalid
    do {
        cout << "Enter test score between 0 and 100: ";
        cin >> s;
    } while (s < 0 && s > 100);

    return s;

};

/**
 * @param arr - An array of integers
 * @param n - The number of elements of the array
 * @pre - The arr is defined and assigned values for all 0 to n - 1 elements
 * @return - The lowest value integer found
 */
int findLowest(int arr[], int n) {

    /*Initialise min to first element of array because you can't have an array with less than
    1 elements
    */
    int
        min = arr[0]
    ;

    // Iterates over n elements from index 1 to index n - 1 and assigns a value lower than min
    to the current element
    for (int i = 1; i < n; i++) {
        if (arr[i] < min) {
            min = arr[i];
        };
    };

    return min;

};

/**
 * @param arr - An array of integers
 * @param n - The number of elements that will be iterated over from index 0 to index n - 1
 * @return - Average float value of elements from index 0 to index n - 1
 */
float calcAverage(int arr[], int n) {

    // Runtime error checking
    if (n <= 1) {
        std::cout
            << "Warning: n is less than or equal to 1, the function can only run with n >= 2
otherwise division "
            << "by 0 or a negative number will take place making the program crash or the
result irrelevant, returning 0.f"
        ;
    }
}
```



```

        ;
        return 0.f;
    };
    int
        total = 0
    ;
    // Add up all integers in arr
    for (int i = 0; i < n; i++) {
        total += arr[i];
    };
    // Subtract the lowest integer from the total
    total -= findLowest(arr, n);
    /*
    Perform implicit float division of the new total and the new total number of scores to
    calculate the average
    */
    return float(total) / (n - 1);
};

/**
 * \param[in] float avg - the average
 * @pre - The actual parameter avg is defined and assigned a value
 * @post - The class avg is output to the console
 */
void displayOutput(float avg) {
    using namespace std;
    // Outputs result to console
    cout
        << "After dropping the lowest test score, the test average is "
        << setprecision(2) << fixed << avg << endl;
};

// Test solution
int main() {
    using namespace std;
    const int
        NUMBER_SCORES = 5
    ;
    // Sets all scores to 0 implicitly
    int
        scores[NUMBER_SCORES] = {}
    ;
    float
        average = 0.f
    ;
    // Format console
    cout
        << endl
        << "Q5 output:" << endl
        << endl
    ;
    // Prompts user to enter scores
    for (int i = 0; i < NUMBER_SCORES; i++) {
        cout << "Score " << i + 1 << ":" << endl;
        scores[i] = getScore();
    };
    // Format console
    cout << endl;
    // Calculates the average and assigns the value to average
    average = calcAverage(scores, NUMBER_SCORES);
    // Outputs class average to console
    displayOutput(average);
    return 0;
};

```

Outputs:

Input data:

65 24 80 73 51

Output:

```
C:\Windows\System32\cmd.e X + v - □ X
(venv) C:\Users\rowan\Desktop\UnisaWork\IntroductionToProgramming1-Cos1511\assessments\Assessment-2>test.bat

Q5 output:

Score 1:
Enter test score between 0 and 100: 65
Score 2:
Enter test score between 0 and 100: 24
Score 3:
Enter test score between 0 and 100: 80
Score 4:
Enter test score between 0 and 100: 73
Score 5:
Enter test score between 0 and 100: 51

After dropping the lowest test score, the test average is 67.25 |
```

Input data:

66 38 84 69 59

Output:

```
C:\Windows\System32\cmd.e X + v - □ X
(venv) C:\Users\rowan\Desktop\UnisaWork\IntroductionToProgramming1-Cos1511\assessments\Assessment-2>test.bat

Q5 output:

Score 1:
Enter test score between 0 and 100: 66
Score 2:
Enter test score between 0 and 100: 38
Score 3:
Enter test score between 0 and 100: 84
Score 4:
Enter test score between 0 and 100: 69
Score 5:
Enter test score between 0 and 100: 59

After dropping the lowest test score, the test average is 69.50 |
```

Input data:

72 52 81 23 53

Output:

```
C:\Windows\System32\cmd.e  X + v - □ X
(venv) C:\Users\rowan\Desktop\UnisaWork\IntroductionToProgramming1-Cos1511\assessments\Assessment-2>test.bat

Q5 output:

Score 1:
Enter test score between 0 and 100: 72
Score 2:
Enter test score between 0 and 100: 52
Score 3:
Enter test score between 0 and 100: 81
Score 4:
Enter test score between 0 and 100: 23
Score 5:
Enter test score between 0 and 100: 53

After dropping the lowest test score, the test average is 64.50
```

Input data:

65 28 72 63 65

Output:

```
C:\Windows\System32\cmd.e  X + v - □ X
(venv) C:\Users\rowan\Desktop\UnisaWork\IntroductionToProgramming1-Cos1511\assessments\Assessment-2>test.bat

Q5 output:

Score 1:
Enter test score between 0 and 100: 65
Score 2:
Enter test score between 0 and 100: 28
Score 3:
Enter test score between 0 and 100: 72
Score 4:
Enter test score between 0 and 100: 63
Score 5:
Enter test score between 0 and 100: 65

After dropping the lowest test score, the test average is 66.25
```

Input data:

65 55 75 68 62

Output:

```
C:\Windows\System32\cmd.e  X  +  v  -  □  X

(venv) C:\Users\rowan\Desktop\UnisaWork\IntroductionToProgramming1-Cos1511\assessments\Assessment-2>test.bat

Q5 output:

Score 1:
Enter test score between 0 and 100: 65
Score 2:
Enter test score between 0 and 100: 55
Score 3:
Enter test score between 0 and 100: 75
Score 4:
Enter test score between 0 and 100: 68
Score 5:
Enter test score between 0 and 100: 62

After dropping the lowest test score, the test average is 67.50
```

Question 6

```
#include <iostream>
#include <string>
/**
 * Prompts user to input height, width and length of the room
 * @param theHeight - integer passed via reference representing the height of the room
 * @param theWidth - integer passed via reference representing the width of the room
 * @param theLength - integer passed via reference representing the length of the room
 * @post theHeight, theWidth and theLength are assigned values based on their corresponding
input
 */
void getData(int& theHeight, int& theWidth, int& theLength) {

    using namespace std;

    // Validate height (must be >= 0)
    do {
        cout << "Enter the height of the room (non-negative): ";
        cin >> theHeight;
        if (theHeight < 0) {
            cout << "Error: Height cannot be negative. Try again." << endl;
        };
    } while (theHeight < 0);

    // Validate width (must be >= 0)
    do {
        cout << "Enter the width of the room (non-negative): ";
        cin >> theWidth;
        if (theWidth < 0) {
            cout << "Error: Width cannot be negative. Try again." << endl;
        };
    } while (theWidth < 0);

    // Validate length (must be >= 0)
    do {
        cout << "Enter the length of the room (non-negative): ";
        cin >> theLength;
        if (theLength < 0) {
            cout << "Error: Length cannot be negative. Try again." << endl;
        };
    } while (theLength < 0);

};

/**
 * Calculates the integer volume of a room
 * @param h - integer representing the height of the room
 * @param w - integer representing the width of the room
 * @param l - integer representing the length of the room
 * @return - the volume of the room
 */
int calculateVolume(const int h, const int w, const int l) {
    // Formula for volume
    return h * w * l;
};

/**
 * Outputs the height, width, length and volume of the room are output to the console in an
organised format
 * @param h - integer representing the height of the room
 * @param w - integer representing the width of the room
 * @param l - integer representing the length of the room
 * @param v - integer representing the length of the room
 * @post - height, width, length and volume of the room are output to the console in an
organised format
```

```

*/
void displayOutput(const int h, const int w, const int l, const int v) {

    using namespace std;

    string
        size = ""
    ;

    // Determines size based on the volume
    if (v < 100) {
        size = "Small";
    } else if (v < 500) {
        size = "Medium";
    } else {
        size = "Large";
    };

    // Outputs text to the console
    cout
        << "The volume of a room with height " << h << ", width "
        << w << " and length " << l << endl
        << "is " << v << ". Size: " << size << endl
    ;

};

// Test solution
int main() {

    using namespace std;

    int
        height = 0,
        width = 0,
        length = 0,
        volume = 0
    ;

    // Runs the main program loop 5 times
    for (int i = 0; i < 5; i++) {

        cout << endl;

        // Prompts user to input data
        getData(height, width, length);

        // Calculates volume and stores it
        volume = calculateVolume(height, width, length);

        // Outputs data to console
        displayOutput(height, width, length, volume);

    };

    // Format console
    cout
        << endl
        << "program finished" << endl
    ;

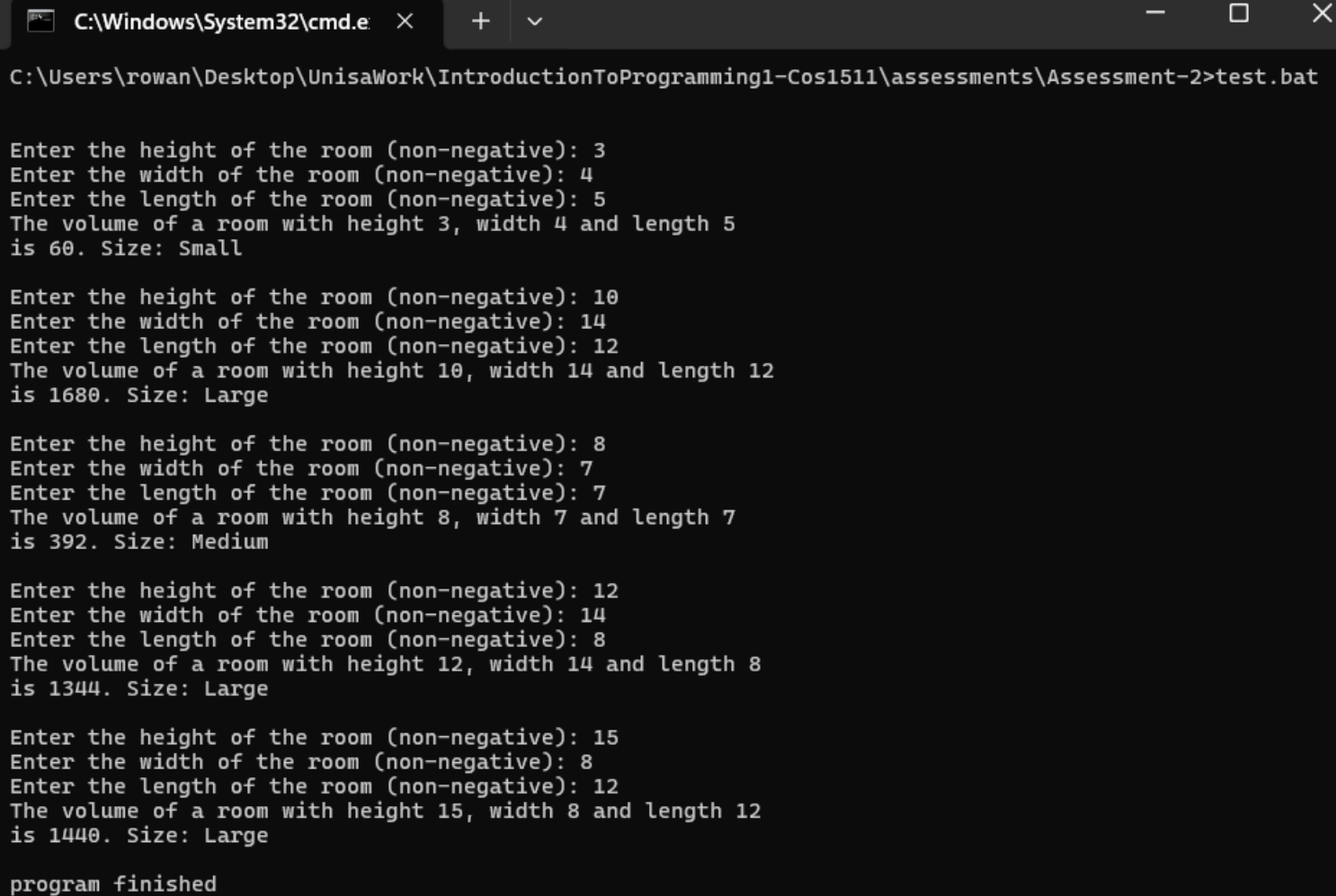
    return 0;

};

```

Input data:

3 4 5
10 14 12
8 7 7
12 14 8
15 8 12

Output:

```
C:\Windows\System32\cmd.e  X + v
C:\Users\rowan\Desktop\UnisaWork\IntroductionToProgramming1-Cos1511\assessments\Assessment-2>test.bat

Enter the height of the room (non-negative): 3
Enter the width of the room (non-negative): 4
Enter the length of the room (non-negative): 5
The volume of a room with height 3, width 4 and length 5
is 60. Size: Small

Enter the height of the room (non-negative): 10
Enter the width of the room (non-negative): 14
Enter the length of the room (non-negative): 12
The volume of a room with height 10, width 14 and length 12
is 1680. Size: Large

Enter the height of the room (non-negative): 8
Enter the width of the room (non-negative): 7
Enter the length of the room (non-negative): 7
The volume of a room with height 8, width 7 and length 7
is 392. Size: Medium

Enter the height of the room (non-negative): 12
Enter the width of the room (non-negative): 14
Enter the length of the room (non-negative): 8
The volume of a room with height 12, width 14 and length 8
is 1344. Size: Large

Enter the height of the room (non-negative): 15
Enter the width of the room (non-negative): 8
Enter the length of the room (non-negative): 12
The volume of a room with height 15, width 8 and length 12
is 1440. Size: Large

program finished
```