UNIVERSITY OF
WEST LONDON

# ENGINEERING SOFTWARE

## Assignment 2 – Functions in C++

Aaron Dellow

21440503

Sound Engineering

# Abstract

The aim of this assignment is to design a program in C++ that can perform a series of different calculations based on a user input. There will be three different calculations that the user can choose, accessible from a recurring user interface (UI). This program will be constructed using functions. This choice is advantageous, as it means the program can run in a less linear way, making for more efficient code. Also, it allows for each calculation to be approached as its own problem that requires a unique solution. To find a solution, flowcharts will be employed before the coding and evidence of testing begins. Once each solution has been created, each function can be employed within the **Main()** to create one overarching program that meets the program requirements.

# Contents

# Introduction

## Program Requirements

- **[1]** All the calculations must be performed within one single program.
- **[2]** The user should be provided with a choice of which calculation to perform.
- **[3]** Prompt the user with an interface to choose the calculation and input the desired values.
- **[4]** The program should run in a continuous manner, returning the user to the interface.
- **[5]** The program must be constructed using functions.

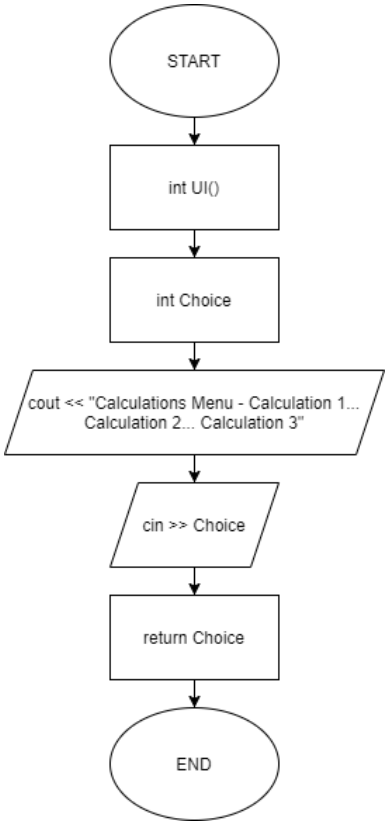
For the program to meet the requirements, appropriate programming practices were utilised. To begin, flowcharts were used to decompose the problem into smaller, more manageable chunks, hence making it easier to begin finding ways to code solutions.

Then, each calculation could be contained within its own function. A function within C++ is a group of statements that perform a task. Even the **Main()** within C++ is a function, which encapsulates the main body of the program's code, always returning zero. Values can be passed into, processed, and returned out of most functions, apart from **void()** functions. Both value-returning and void functions will be used. Variable referencing, a technique used within functions to alter a variable's stored value instead of copying it in, will also be used. In terms of code layout, function prototyping will be used. Using function prototypes allows for the **Main()** to be the first function that you see, with simple initialisations of all the tributary functions above it. The actual functions can be declared after the **Main()**, providing much neater code that is easier to look at and read from.

To allow the user's decision making to be reflected in the program behaviour, conditional statements must be used. In this specific case, if statements and while loops allow for the code to act in a more non-linear fashion and 'make decisions' based on the user's input. While loops were also used to allow the code to run in a continuous manner, as per Program Requirement **[4]**.

# Methodology

## Function 1 – UI()




*Flowchart 1 - UI()*

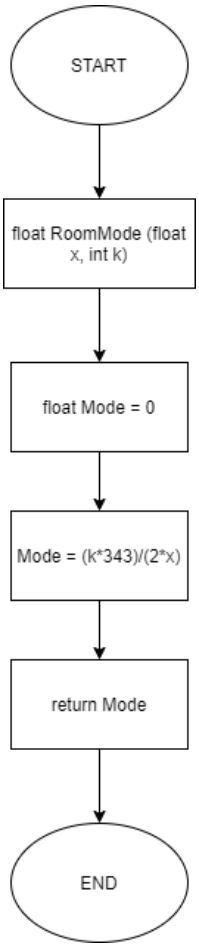
## Code Description/Analysis

**UI()** is a value-returning function that outputs the user interface and takes in the user's input. Although no values need to be passed in, a value-returning function is used so that a variable's value can be returned for use within the **Main()**. A local variable, **Choice**, is used to achieve this. The output statement on Line 111 has been cut off in the screenshot below but can be seen in the C++ Code chapter.

```
107     // 1. User Interface
108     int UI()
109     {
110         int Choice; // local variable Choice declared
111         cout << "\n" << "-Calculations Menu-" << "\n" << "1. Room Mode Calculator" << "\n"
112         cout << "Select an option: ";
113         cin >> Choice; // user inputs choice
114         cout << "\n";
115         return Choice;
116     }
```

*Figure 1 - UI() Code*

## Function 2 – RoomModes()




*Flowchart 2 - RoomModes()*

$$\boldsymbol{f_k = k * (c \,/\, (2L))}$$

*Equation 1 - Axial Room Mode Formula*

$f_k$ = Room Mode Frequency

$k$ = Mode Number

$c$ = Speed of Sound (343m/s)

$L$ = Distance between boundary surfaces

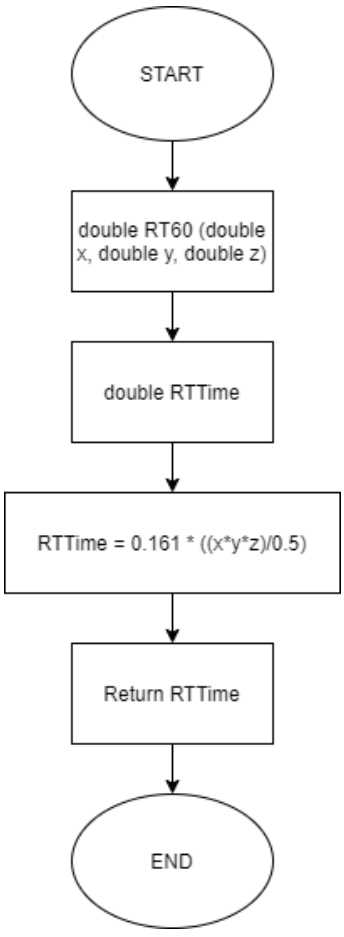
Code Description/Analysis

**RoomModes()** is a value-returning float function that has two global variables passed into it. A float data type was chosen as the function is returning a frequency value in Hertz, which are often measured to the precision of one or two decimal places. Float variable **Mode** is used to perform the calculation (see Equation 1 above) and return the result to the **Main()**.

```
118     // 2. Room Mode Calculator
119     float RoomModes(float x, int k) // two global variables passed in
120     {
121         float Mode = 0;
122         Mode = (k * 343) / (2 * x); // room mode calculation carried out
123         return Mode;
124     }
```

*Figure 2 - RoomModes() Code*

## Function 3 – RT60()




*Flowchart 3 - RT60()*

$$\boldsymbol{RT60 = k * (V \,/\, Sa)}$$

*Equation 2 - Reverberation Time Formula*

$RT60$ = Reverberation Time

$k$ = RT60 constant (0.161 for metric system)

$V$ = Total volume of room

$Sa$ = Total surface absorption coefficient = 0.5

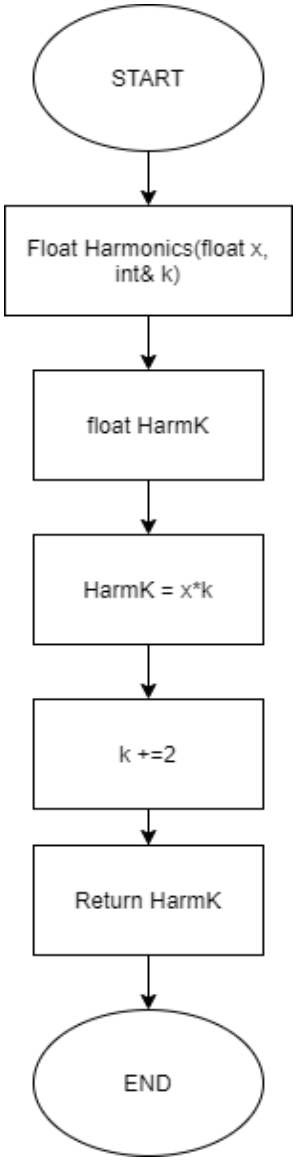
## Code Description/Analysis

**RT60()** is a double function. When first attempted, this function used the float data type. However, when debugging, an error was received due to the mathematical processing of variables **x**, **y**, and **z** on Line 130. Changing the data types to double solved this issue. Local variable **RTTime** is used to return the calculated value. See Equation 2 above for explanation of its variables. Variable **V** seen in the formula above is represented by three dimension values that the user provides.

```
126     // 3. RT60 Calculator
127     double RT60(double x, double y, double z) // three global variables passed in
128     {
129         double RTTime;
130         RTTime = 0.161 * ((x*y*z) / 0.5); //RT60 calculation carried out
131         return RTTime;
132     }
```

*Figure 3 - RT60() Code*

## Function 4 – Harmonics()




*Flowchart 4 - Harmonics()*

$$f_k = f_1 * k$$

*Equation 3 - Frequency Harmonics Formula*

$f_1$ = Fundamental frequency / First harmonic

$k$ = Number of the harmonic

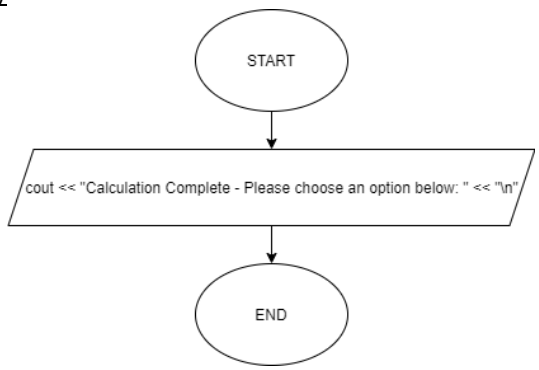
## Code Description/Analysis

**Harmonics()** is a float function. One float variable is passed in, and an int variable is referenced into the function. Equation 3, seen above, is seen on Line 138. On line 139, the value of **k** is increased by two. This variable is used to iterate a while loop that this function is nested in. Variable **HarmK** is returned to the **Main()**. Value-returning functions are the obvious choice for all the calculation functions, mainly for their call-and-return abilities.

```
134     // 4. Room Harmonics Calculator
135     float Harmonics(float x, int& k) // One variable passed in, one variable referenced
136     {
137         float HarmK;
138         HarmK = x * k; // Harmonics calculation - frequency * order of harmonic
139         k += 2; // 2 added to harmonic order no. so that function can be used to iterate with
140         return HarmK;
141     }
```

*Figure 4 - Harmonics() Code*

# Function 5 – Complete()




*Flowchart 5 - Complete()*

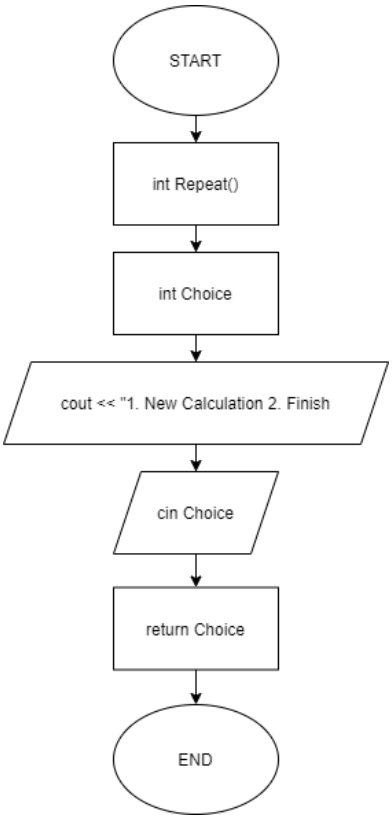
## Code Description/Analysis

**Complete()** is a simple void function, used to print a message once a calculation has been completed.

```
143     // 5. Completion Message
144     void Complete() // used after any calculation has been completed
145     {
146         cout << "\n" << "Calculation complete! Please choose an option below: " << "\n";
147     }
```

*Figure 5 - Complete() Code*

## Function 6 – Repeat()




*Flowchart 6 - Repeat()*

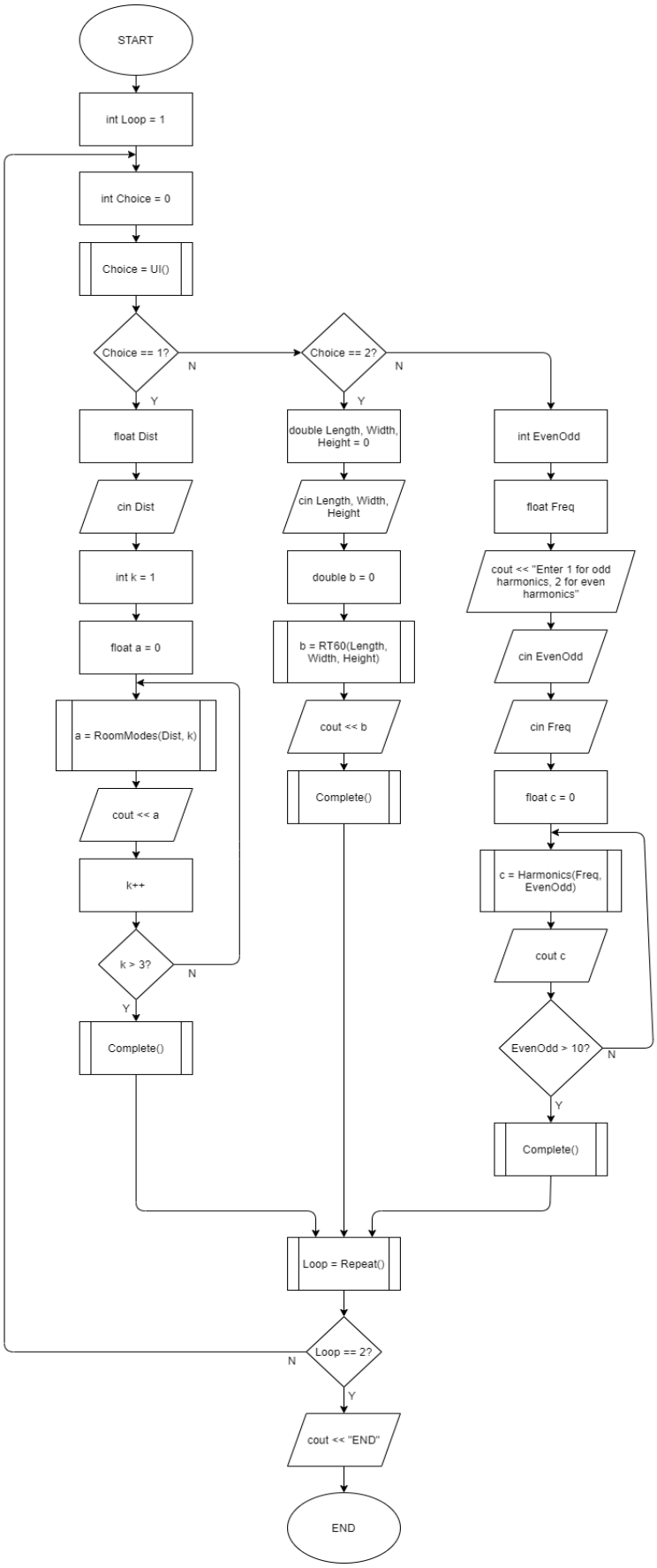
## Code Description/Analysis

**Repeat()** is a function that runs after the **Complete()** function when a calculation has been completed. It outputs 2 options for the user and stores their choice in a local variable. The input is then returned.

```
149     // 6. Repeat Function
150     int Repeat()
151     {
152         int Choice;
153         cout << "1. New Calculation 2. Finish: " << "\n"; // options printed for user
154         cin >> Choice;
155         return Choice; // chosen option returned
156     }
```

*Figure 6 - Repeat() Code*

## Function 7 – Main()




*Flowchart 7 - Main()*

## Code Description/Analysis

Firstly, all the functions are prototyped (Lines 6-11) after the pre-program directives (Lines 1-2).

```
1   #include <iostream>
2   using namespace std;
3
4   // prototypes for all functions declared before main()
5
6   int UI();
7   float RoomModes(float x, int k);
8   double RT60(double x, double y, double z);
9   float Harmonics(float x, int& k);
10  void Complete();
11  int Repeat();
```

*Figure 7 - Pre-Program Directives and Function Prototypes*

Next, the **Main()** begins. A variable **Loop** is initialised and a while loop begins on Line 18. This loop keeps the program running in a continuous manner. The main body of the code exists within this loop. Next, a global variable **Choice** is initialised to call the **UI()** function, which returns the user's selection. Then, a series of conditional statements are used to decide which calculation needs to be performed.

```
15  int main()
16  {
17      int Loop = 1; // variable is used for the while loop
18      while (Loop == 1)
19      {
20          int Choice = 0;
21          while (Choice < 1 or Choice > 3) // validating user input
22          {
23              Choice = UI(); // running UI function
24          }
25
26          if (Choice == 1) { ... }
46
47          else if (Choice == 2) { ... }
70
71          else { ... }
97
98          Loop = Repeat();
99      }
100     cout << "END";
101
102     return 0;
103 }
```

*Figure 8 - Main() - Condensed*

Lines 26-45 contains the code which calculates the first three room modes when given a distance value in meters. Float variable **Dist** is declared on Line 30, which is used to take an input for the distance in meters. Then, **k** is initialised on Line 35. This is used to iterate the while loop on Line 37. Float variable **a** is initialised to call the **RoomModes()** function. As three room modes need to be calculated, the while loop is set up to iterate three times before exiting the loop. Within the loop, function **RoomModes()** is called using **a**, the output is printed to the user, and one is added to **k**.

```
26      if (Choice == 1) // selection 1 - room mode calculator
27      {
28          cout << "You have selected: Room Mode Calculator" << "\n";
29
30          float Dist;
31          cout << "Please enter the distance between the boundary surfaces of the room in me
32          cin >> Dist; // user inputs length/width in meters
33          cout << "\n";
34
35          int k = 1; // used to iterate while loop and print room mode number
36          float a = 0; // used to call RoomModes function
37          while (k < 4)
38          {
39              a = RoomModes(Dist, k);
40              cout << "Room mode " << k << " is " << a << "Hz." << "\n"; // output printed
41              k++;
42          }
43
44          Complete();
```

*Figure 9 - Room Modes Calculation*

Once **k** is no longer less than four, the loop terminates. Then, the **Complete()** function is run on line 44.

Lines 47-69 contains the code which calculates the RT60 time of a room. Variables **Length**, **Width**, and **Height** are initialised on Line 51, then inputs are received from the user. On Line 63, variable **b** is initialised, which is used to call the **RT60()** function on Line 64. Then, the output is printed, and the **Complete()** function is run on Line 67.

```
47      else if (Choice == 2)
48      {
49          cout << "You have selected: RT60 Calculator" << "\n";
50
51          double Length, Width, Height = 0;
52          cout << "Please input the length, then the width, then the height of the room in
53
54          cout << "L: ";
55          cin >> Length;
56
57          cout << "W: ";
58          cin >> Width;
59
60          cout << "H: ";
61          cin >> Height; // user inputs room dimensions
62
63          double b = 0;
64          b = RT60(Length, Width, Height); // variable b used to call RT60 function
65          cout << "\n" << "The RT60 time for the room is " << b << " seconds." << "\n";
66
67          Complete();
68
69      }
```

*Figure 10 - RT60 Calculation*

Lines 71-96 contains the code which calculates the odd or even harmonics of a frequency. Two variables are declared; **EvenOdd** is used to calculate the even or odd harmonics, while **Freq** lets the user input a frequency in Hertz. On Line 77, the user's input for **EvenOdd** is validated, and then the frequency is inputted by the user on Line 84. Float variable **c** is initialised. The function **Harmonics()** is run using **c**, nested inside a while loop on Line 89. The while loop iterates five times, printing a value each time, so that the program can calculate the first five odd or even harmonics. Finally, the **Complete()** function is run on Line 95.

```
71      else
72      {
73          cout << "You have selected: Even/Odd Harmonics Calculator" << "\n";
74          int EvenOdd = 0;
75          float Freq; // declaring variables to pass into function
76
77          while (EvenOdd < 1 or EvenOdd > 2) // validates user input
78          {
79              cout << "Enter 1 to calculate odd harmonics, enter 2 to calculate even harmonics: ";
80              cin >> EvenOdd; // user inputs 1 or 2
81          }
82
83          cout << "Enter the frequency of the first harmonic in Hertz: ";
84          cin >> Freq; // user inputs first harmonic
85          cout << "\n";
86
87          float c = 0;
88
89          while (EvenOdd < 11) // iterates function
90          {
91              c = Harmonics(Freq, EvenOdd);
92              cout << "Harmonic " << (EvenOdd-2) << " is " << c << " Hz." << "\n"; // outputs result
93          }
94
95              Complete();
96      }
```

*Figure 11 - Odd/Even Harmonics Calculation*

Next, the **Loop** variable that was initialised at the start of the program is used to call the **Repeat()** function on Line 98, which allows the program flow to exit the while loop that was entered on Line 18. Depending on the returned value from the **Repeat()** function, the code will either exit the loop (end of program) or loop back again (new calculation).

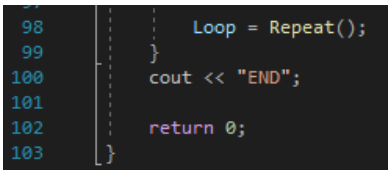



*Figure 12 - Repeat() and End of Program*

## Evidence Of Testing

In this example, variable **a** was creating an error message, as it had been initialised with the **int** data type (Line 35), but was calling **RoomModes()**, a function with the data type **float** (Line 57). Once **a** had been changed to a **float** variable, the error message disappeared.

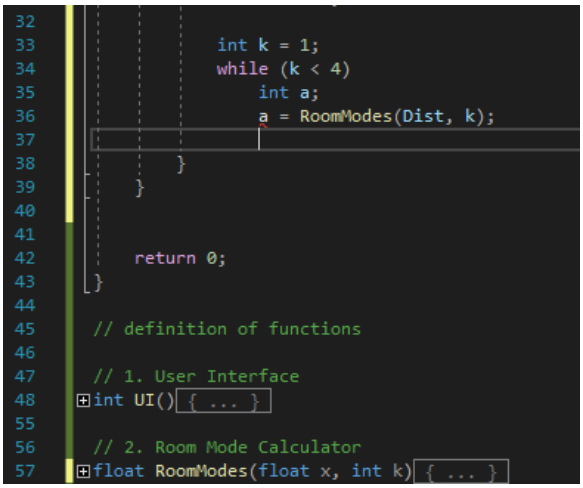



*Figure 13 - Incorrect Variable Data Type*

The next error encountered was incorrect printing. Instead of iterating up through the harmonics values, the program was printing the first harmonic over and over.

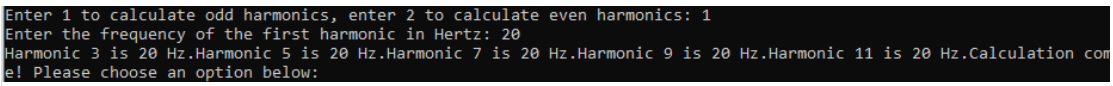



*Figure 14 - Harmonics Printing Error*

On further inspection of the code, the **Freq** variable was being used in the printing message, rather than the variable **c**. Once this had been rectified, the code ran successfully.

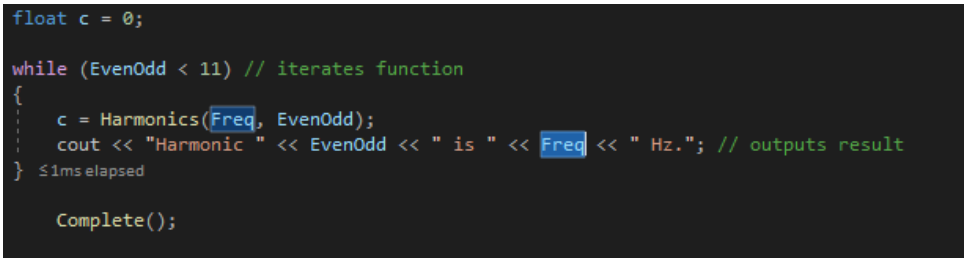



*Figure 15 - Incorrect Harmonics Code*

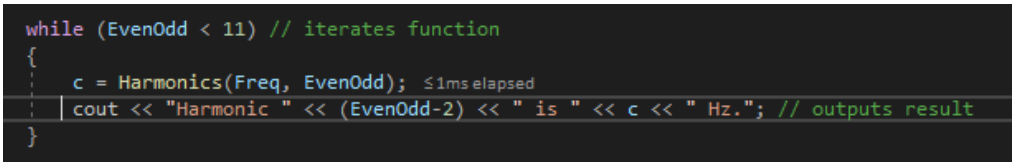



*Figure 16 - Corrected Code*

## Successful Test of Program

```
You have selected: Room Mode Calculator
Please enter the distance between the boundary surfaces of the room in meters (i.e. length/width): 6

Room mode 1 is 28.5833Hz.
Room mode 2 is 57.1667Hz.
Room mode 3 is 85.75Hz.

Calculation complete! Please choose an option below:
1. New Calculation 2. Finish:
1
```

*Figure 17 - Room Modes Calculation*

```
You have selected: RT60 Calculator
Please input the length, then the width, then the height of the room in meters:

L: 4
W: 6
H: 3

The RT60 time for the room is 23.184 seconds.

Calculation complete! Please choose an option below:
1. New Calculation 2. Finish:
1
```

*Figure 18 - RT60 Calculation*

```
You have selected: Even/Odd Harmonics Calculator
Enter 1 to calculate odd harmonics, enter 2 to calculate even harmonics: 1
Enter the frequency of the first harmonic in Hertz: 30

Harmonic 1 is 30 Hz.
Harmonic 3 is 90 Hz.
Harmonic 5 is 150 Hz.
Harmonic 7 is 210 Hz.
Harmonic 9 is 270 Hz.

Calculation complete! Please choose an option below:
1. New Calculation 2. Finish:
1
```

*Figure 19 - Odd Harmonics Calculation*

```
You have selected: Even/Odd Harmonics Calculator
Enter 1 to calculate odd harmonics, enter 2 to calculate even harmonics: 2
Enter the frequency of the first harmonic in Hertz: 30

Harmonic 2 is 60 Hz.
Harmonic 4 is 120 Hz.
Harmonic 6 is 180 Hz.
Harmonic 8 is 240 Hz.
Harmonic 10 is 300 Hz.

Calculation complete! Please choose an option below:
1. New Calculation 2. Finish:
2
```

*Figure 20 - Even Harmonics Calculation*

# Conclusion

Overall, this assignment was carried out successfully. The created program successfully meets the requirements: a recurring user interface that takes the user's input; code constructed with functions; code that runs in a continuous manner, all within one single program. The problem was successfully analysed and decomposed using flowcharts, which allowed for easy construction of the code. With some debugging, the code was optimised to run smoothly and efficiently. After this, it successfully carries out the required operations.

# C++ Code

```
#include <iostream>
using namespace std;

// prototypes for all functions declared before main()

int UI();
float RoomModes(float x, int k);
double RT60(double x, double y, double z);
float Harmonics(float x, int& k);
void Complete();
int Repeat();

// main() begins

int main()
{
    int Loop = 1; // variable is used for the while loop
    while (Loop == 1)
    {
        int Choice = 0;
        while (Choice < 1 or Choice > 3) // validating user input
        {
            Choice = UI(); // running UI function
        }

        if (Choice == 1) // selection 1 - room mode calculator
        {
            cout << "You have selected: Room Mode Calculator" << "\n";

            float Dist;
            cout << "Please enter the distance between the boundary surfaces of the
room in meters (i.e. length/width): ";
            cin >> Dist; // user inputs length/width in meters
            cout << "\n";

            int k = 1; // used to iterate while loop and print room mode number
            float a = 0; // used to call RoomModes function
            while (k < 4)
            {
                a = RoomModes(Dist, k);
                cout << "Room mode " << k << " is " << a << "Hz." << "\n"; // output
printed
                k++;
            }

            Complete();
        }

        else if (Choice == 2)
        {
            cout << "You have selected: RT60 Calculator" << "\n";

            double Length, Width, Height = 0;
            cout << "Please input the length, then the width, then the height of the
room in meters: " << "\n" << "\n";

            cout << "L: ";
            cin >> Length;
```

```
            cout << "W: ";
            cin >> Width;

            cout << "H: ";
            cin >> Height; // user inputs room dimensions

            double b = 0;
            b = RT60(Length, Width, Height); // variable b used to call RT60 function
            cout << "\n" << "The RT60 time for the room is " << b << " seconds." <<
"\n";

            Complete();

        }

        else
        {
            cout << "You have selected: Even/Odd Harmonics Calculator" << "\n";
            int EvenOdd = 0;
            float Freq; // declaring variables to pass into function

            while (EvenOdd < 1 or EvenOdd > 2) // validates user input
            {
                cout << "Enter 1 to calculate odd harmonics, enter 2 to calculate even
harmonics: ";
                cin >> EvenOdd; // user inputs 1 or 2
            }

            cout << "Enter the frequency of the first harmonic in Hertz: ";
            cin >> Freq; // user inputs first harmonic
            cout << "\n";

            float c = 0;

            while (EvenOdd < 11) // iterates function
            {
                c = Harmonics(Freq, EvenOdd);
                cout << "Harmonic " << (EvenOdd-2) << " is " << c << " Hz." << "\n";
// outputs result
            }

                Complete();
        }

        Loop = Repeat();
    }
    cout << "END";

    return 0;
}

// definition of functions

// 1. User Interface
int UI()
{
    int Choice; // local variable Choice declared
    cout << "\n" << "-Calculations Menu-" << "\n" << "1. Room Mode Calculator" << "\n"
<< "2. RT60 Calculator" << "\n" << "3. Even/Odd Harmonics Calculator" << "\n" << "\n";
// UI outputted
    cout << "Select an option: ";
```

```
    cin >> Choice; // user inputs choice
    cout << "\n";
    return Choice;
}

// 2. Room Mode Calculator
float RoomModes(float x, int k) // two global variables passed in
{
    float Mode = 0;
    Mode = (k * 343) / (2 * x); // room mode calculation carried out
    return Mode;
}

// 3. RT60 Calculator
double RT60(double x, double y, double z) // three global variables passed in
{
    double RTTime;
    RTTime = 0.161 * ((x*y*z) / 0.5); //RT60 calculation carried out
    return RTTime;
}

// 4. Room Harmonics Calculator
float Harmonics(float x, int& k) // One variable passed in, one variable referenced
{
    float HarmK;
    HarmK = x * k; // Harmonics calculation - frequency * order of harmonic
    k += 2; // 2 added to harmonic order no. so that function can be used to iterate
with
    return HarmK;
}

// 5. Completion Message
void Complete() // used after any calculation has been completed
{
    cout << "\n" << "Calculation complete! Please choose an option below: " << "\n";
}

// 6. Repeat Function
int Repeat()
{
    int Choice;
    cout << "1. New Calculation 2. Finish: " << "\n"; // options printed for user
    cin >> Choice;
    return Choice; // chosen option returned
}
```