Functions

The basic building block for C programs is functions. A function can take arguments (or not) and return a value (or not.) Every function requires a prototype (declaration) and definition. A prototype tells the C compiler the input and output types of a function are. In this following example demonstrates functions:

```
// Return the sum of two integer
int add_two_numbers(int a, int b)
{
        return (a+b);
}
```

The prototype is the function declaration followed by a semicolon and will appear in a header file or near the top of a C file:

```
int add_two_numbers(int a, int b);
```

The declaration states that the function name is `add_two_numbers()` and it requires two integers to be passed to it and returns an integer. The `return` statement (not a function) returns to the calling function and gives the result of the function. Putting it all together, for example:

```
#include <stdio.h>

// Place function prototypes here
int add_two_numbers(int a, int b);

int main(int argc, char * argv[])
{
        int i;
        int sum;

        i=5;
        // Add two numbers & print result
        sum=add_two_numbers(10,i);
        printf("The sum of %d and 10 is %d\n",I,sum);
        return 0;
}

// Add two integers
int add_two_numbers(int a, int b)
{
        return (a+b);
}
```

The return at the end of `main()` is required (for good coding practice) because `main()` was declared to return an `int`. Returning zero indicates that the program ran successfully.

Functions are used to naturally organize code for ease of coding and ease of others that read to code. Thus functions can drastically reduces the size of the program.

For Loops

The for loop is a basic way to loop a number of times through the same code. The syntax for the for loop is as follows:

```
for(initial condition; loop while true condition; iteration event) {
        <code>
}
```

For example, to count to 10:

```
int temp;

for (temp=1;temp<11;temp++) {
        printf("%d\n",temp);
}
```

`temp` starts at one and is incremented each time thought the loop (`temp++`). Once `temp<11` is not true, the loop breaks. In each iteration the value of `temp` is printed.

Write a program that calculates z for $z^2=x^2+y^2$ given x and y. Use the function `sqrt()` to take the square root and be sure to `#include <math.h>` and link the math library (`-lm`). Print the values from -1.0 to 1.0 for x and y in steps of 0.1 using a nested for loop. Make the calculation a function. The loop should be in `main()`.

```
# cc my_prog.c -lm
```