CS270 Recitation 1
"C Programming Exercise"

Goals

To modify, compile, and run a C program that solves quadratic equations.

The Assignment

Start by making a directory called cs270 in your home directory

```
%> mkdir cs270
%> cd cs270
```

Make a subdirectory called R1 (inside cs270) for the recitation assignment, all files should reside in this subdirectory.

```
%> mkdir R1
%> cd R1
```

We have provided the framework of the C program to get you started. Open up `gedit` (or text editor of your preference), copy the code shown below, and save it into a file called r1.c in your R1 subdirectory.

```
#include <stdlib.h> // for exit()
#include <stdio.h>  // for stderr and fprintf()

// Function: isBitSet
// Description: Determines if a particular bit position is set in a 32 bit uint.
// Return: 0 if the specified bit position is not set (bit equal to 0)
//         and a number other than zero if the bit position is set (bit equal to 1).
// Errors: The bit position should have a value between 0 and 31 inclusive.
unsigned int isBitSet(unsigned int bit_pattern, unsigned int bit_position)
{
    // Exit the program if the bit_position is not between 0 and 31 inclusive.
    // Don't have to check lower bound because the number is unsigned.
    if (bit_position >= 32) {
        fprintf(stderr, "ERROR: bit_position is not in 0 to 31 range.");
        exit(0);
    }

    // Create a mask at the specified bit position.
        // TODO: Implement

    // Check if the specified bit is set and return a non-zero value if it is.
    // Return a zero value if the bit is NOT set.
        // TODO: Implement

    return 0;  // initial implementation always returns 0
}

// Program entry point
int main()
{
    unsigned int bit_pattern, bit_position, bit_set;

    // Query user for the bit pattern and which bit position to check.
    printf ("Bit Set Program\n");
    printf("Enter a hexadecimal number (e.g., 0xFEED1DAD): ");
    scanf("%x", &bit_pattern);
    printf("Enter a bit position (MSB is 31, LSB is 0): ");
    scanf("%d", &bit_position);
    // Call function to determine if the bit is set.
    bit_set = isBitSet(bit_pattern, bit_position);

    // Note that C assumes anything other than zero is true.
    if (bit_set) {
        printf("In 0x%X, bit position %d is set\n", bit_pattern, bit_position);
    } else {
        printf("In 0x%X, bit position %d is not set\n",
            bit_pattern, bit_position);
    }
}
```

Compile the program into an executable called r1, as shown below.

```
%> gcc -g -std=c99 -Wall r1.c -o r1
```

To run the compiled program, type the following command:

```
%> ./r1
```

Verify that the program always returns zero for whether the bit is set even if the bit pattern 0xFFFFFFFF is used. This happens because the isBitSet() function has not been implemented. Edit the program (using `gedit, or any other editor`) and implement the isBitSet() function in r1.c. Recompile and run the program with the following test sets:

Bit pattern = 0x1, bit_position = 0, should print "In 0x1, bit position 0 is set"
Bit pattern = 0x1, bit_position = 5, should print "In 0x1, bit position 5 is not set"
Bit pattern = 0xFEED1DAD, bit_position = 27, should print "In 0xFEED1DAD, bit position 27 is set"