Thomas Nguyen
Professor Vu
COEN 171

Homework #7

<u>Problem Set</u>

8.1 Describe three situations where a combined counting and logical looping statement is needed.

1. You're using a loop to iteratively add values to a certain variable. However, you need to exit the loop if the variable hits / exceeds a certain value.
2. You're iteratively looking for an item in an array or linked list and you need to exit right when the item is found.
3. If you're moving items from a linked list over to an array, you need to stop moving values over from one source to the other when the linked list ends or when the array is full.

8.4 What are the pros and cons of using unique closing reserved words on compound statements?

Pros:

- Increased readability (it is clear when a certain block of code ends)

Cons:

- Less orthogonality because of the increase in keywords

8.9 What are the arguments both for and against the exclusive use of Boolean expressions in the control statements in Java (as opposed to also allowing arithmetic expressions, as in C++)?

Only allowing the use of Boolean expressions in control statements increases the readability and reliability of the code. There is no confusion (as there is in C++) as to what a control statement may evaluate to because it is a universal Boolean expression. It also ensures that there are no references to variables of incorrect types.

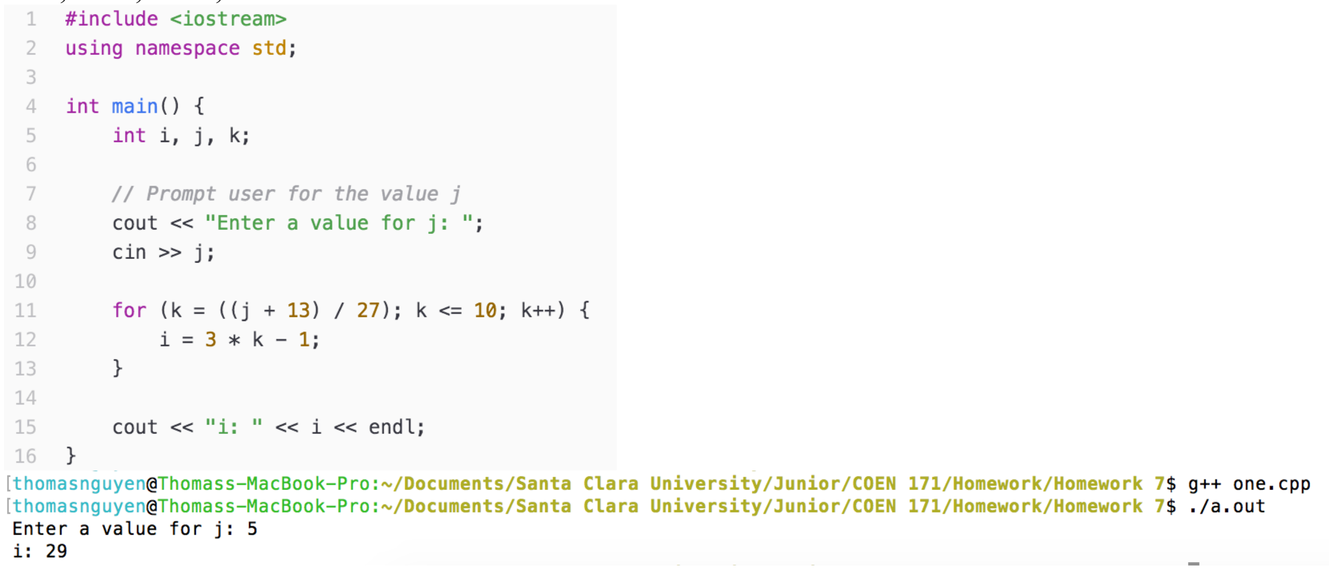
8.11 Describe three specific programming situations that require a posttest loop

1. When you need the user program to run at least once.
2. When you need the user input before continuing the program.
3. If you are traversing a linked list and want to check if you have pointed to null before trying to access memory.

Programming Exercises

8.1 Rewrite the following pseudocode segment using a loop structure in the specified languages:

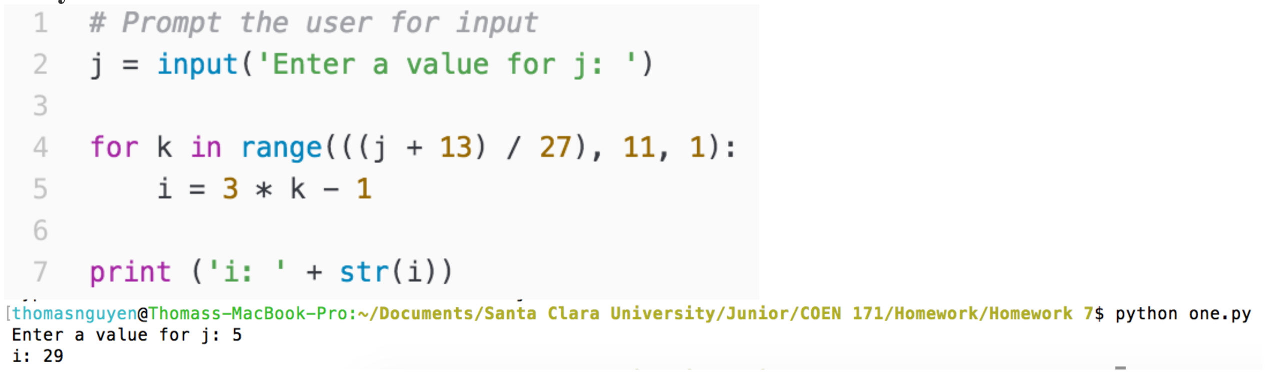
a. C, C++, Java, or C#



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

int main() {
    int i, j, k;

    // Prompt user for the value j
    cout << "Enter a value for j: ";
    cin >> j;

    for (k = ((j + 13) / 27); k <= 10; k++) {
        i = 3 * k - 1;
    }

    cout << "i: " << i << endl;
}
```

```
[thomasnguyen@Thomass-MacBook-Pro:~/Documents/Santa Clara University/Junior/COEN 171/Homework/Homework 7$ g++ one.cpp
[thomasnguyen@Thomass-MacBook-Pro:~/Documents/Santa Clara University/Junior/COEN 171/Homework/Homework 7$ ./a.out
Enter a value for j: 5
i: 29
```

b. Python



```python
# Prompt the user for input
j = input('Enter a value for j: ')

for k in range(((j + 13) / 27), 11, 1):
    i = 3 * k - 1

print ('i: ' + str(i))
```

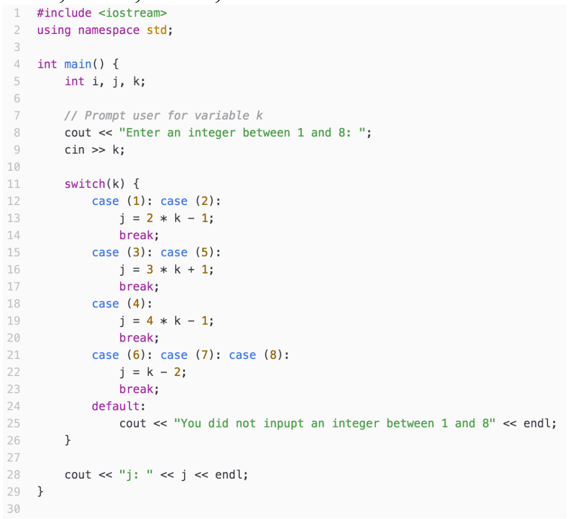
```
[thomasnguyen@Thomass-MacBook-Pro:~/Documents/Santa Clara University/Junior/COEN 171/Homework/Homework 7$ python one.py
Enter a value for j: 5
i: 29
```

Assume all variables are integer type. Discuss which language, for this code, has the best writability, the best readability, and the best combination of the two.

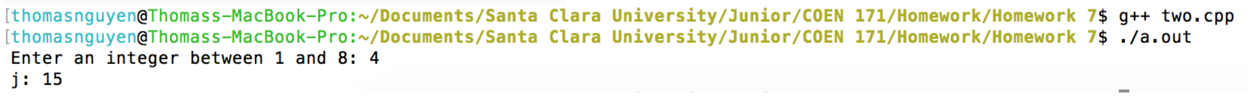
As seen from above, Python easily has the best writability because of its quick and easy implementation. It is also very clear in its readability when asking for input and printing to its output. One issue I found in its readability though is its for loop. It could be because the first language I learned was C, but it is not easily apparent in Python that we are looping over a range object that we created. I think if C++ had a simpler iostream library, this would be the best combination of the two.

8.3 Rewrite the following code segment using a multiple-selection statement in the following languages:

a. C, C++, Java, or C#

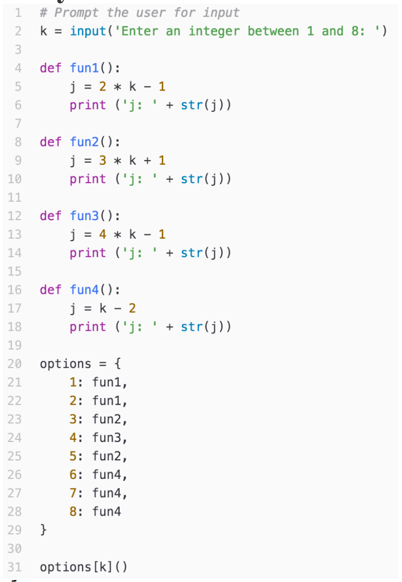


```cpp
#include <iostream>
using namespace std;

int main() {
    int i, j, k;

    // Prompt user for variable k
    cout << "Enter an integer between 1 and 8: ";
    cin >> k;

    switch(k) {
        case (1): case (2):
            j = 2 * k - 1;
            break;
        case (3): case (5):
            j = 3 * k + 1;
            break;
        case (4):
            j = 4 * k - 1;
            break;
        case (6): case (7): case (8):
            j = k - 2;
            break;
        default:
            cout << "You did not inpupt an integer between 1 and 8" << endl;
    }

    cout << "j: " << j << endl;
}
```

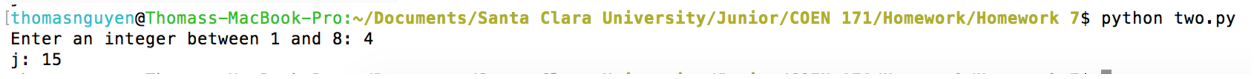


```
[thomasnguyen@Thomass-MacBook-Pro:~/Documents/Santa Clara University/Junior/COEN 171/Homework/Homework 7$ g++ two.cpp
[thomasnguyen@Thomass-MacBook-Pro:~/Documents/Santa Clara University/Junior/COEN 171/Homework/Homework 7$ ./a.out
 Enter an integer between 1 and 8: 4
 j: 15
```

b. Python



```python
# Prompt the user for input
k = input('Enter an integer between 1 and 8: ')

def fun1():
    j = 2 * k - 1
    print ('j: ' + str(j))

def fun2():
    j = 3 * k + 1
    print ('j: ' + str(j))

def fun3():
    j = 4 * k - 1
    print ('j: ' + str(j))

def fun4():
    j = k - 2
    print ('j: ' + str(j))

options = {
    1: fun1,
    2: fun1,
    3: fun2,
    4: fun3,
    5: fun2,
    6: fun4,
    7: fun4,
    8: fun4
}

options[k]()
```

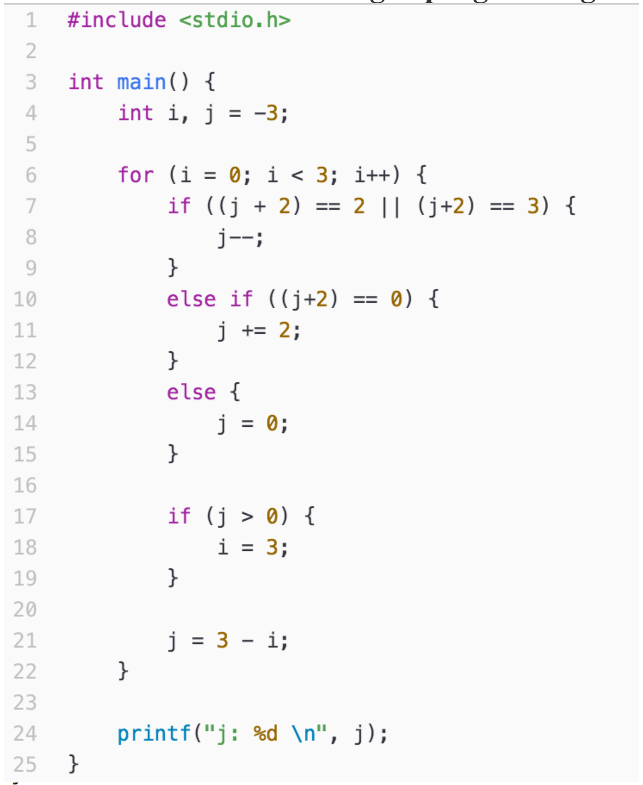


```
[thomasnguyen@Thomass-MacBook-Pro:~/Documents/Santa Clara University/Junior/COEN 171/Homework/Homework 7$ python two.py
 Enter an integer between 1 and 8: 4
 j: 15
```

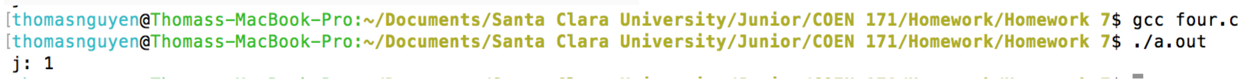
Assume all variables are integer type. Discuss the relative merits of the use of the languages for this particular code.

C++ actually has a defined switch-case functionality while in Python we needed to create our own switch-case functionality. Switch cases can often be useful, so I find it important that C++ has the switch case option.

8.4 Consider the following C program segment. Rewrite it using no gotos or breaks.

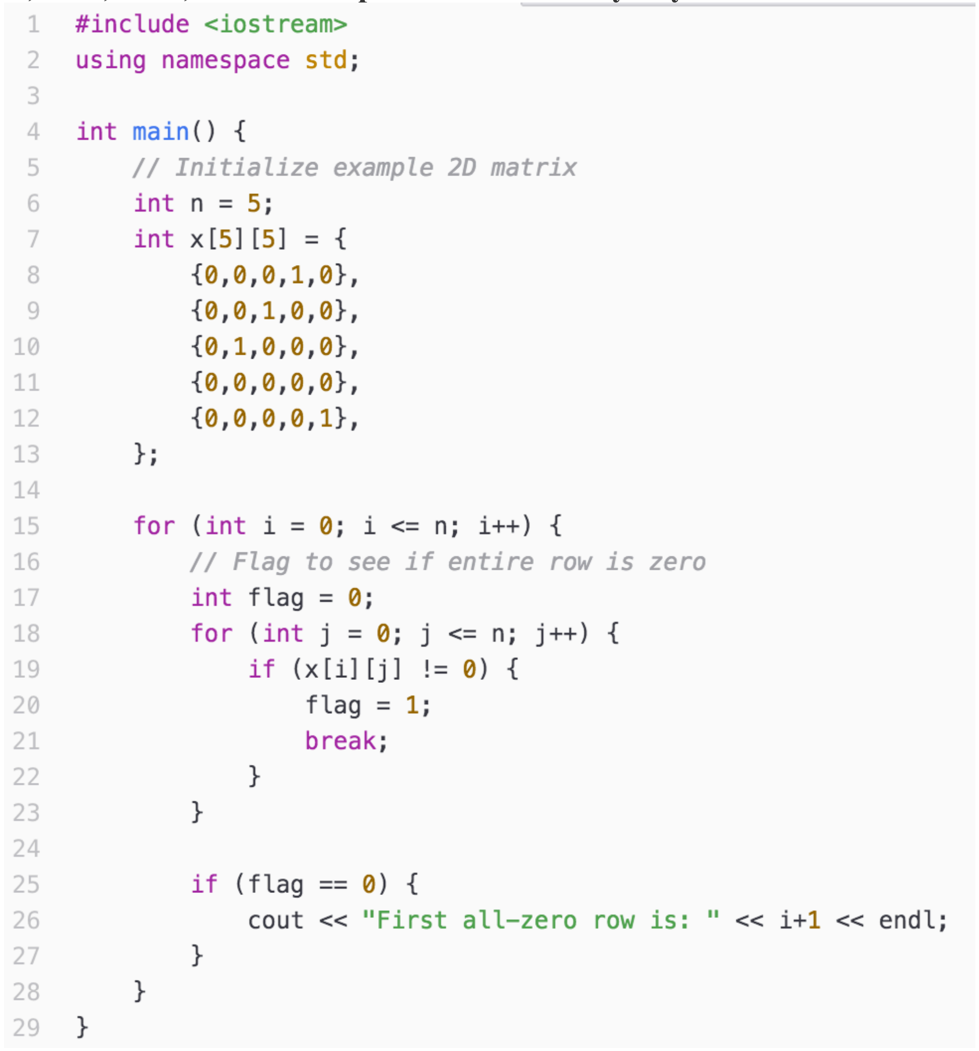


```c
#include <stdio.h>

int main() {
    int i, j = -3;

    for (i = 0; i < 3; i++) {
        if ((j + 2) == 2 || (j+2) == 3) {
            j--;
        }
        else if ((j+2) == 0) {
            j += 2;
        }
        else {
            j = 0;
        }

        if (j > 0) {
            i = 3;
        }

        j = 3 - i;
    }

    printf("j: %d \n", j);
}
```

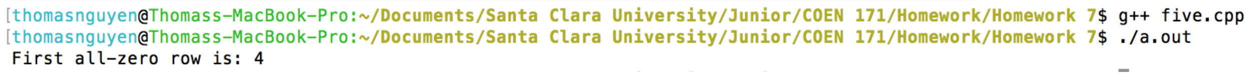


```
[thomasnguyen@Thomass-MacBook-Pro:~/Documents/Santa Clara University/Junior/COEN 171/Homework/Homework 7$ gcc four.c
[thomasnguyen@Thomass-MacBook-Pro:~/Documents/Santa Clara University/Junior/COEN 171/Homework/Homework 7$ ./a.out
j: 1
```

8.5 In a letter to the editor of CACM, Rubin (1987) uses the following code segment as evidence that the readability of some code with gotos is better than the equivalent code without gotos. This code finds the first row of an n by n integer matrix named x that has nothing but zero values. Rewrite this code without gotos in one of the following languages: C, C++, Java, or C#. Compare the readability of your code to that of the example code.



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

int main() {
    // Initialize example 2D matrix
    int n = 5;
    int x[5][5] = {
        {0,0,0,1,0},
        {0,0,1,0,0},
        {0,1,0,0,0},
        {0,0,0,0,0},
        {0,0,0,0,1},
    };

    for (int i = 0; i <= n; i++) {
        // Flag to see if entire row is zero
        int flag = 0;
        for (int j = 0; j <= n; j++) {
            if (x[i][j] != 0) {
                flag = 1;
                break;
            }
        }

        if (flag == 0) {
            cout << "First all-zero row is: " << i+1 << endl;
        }
    }
}
```



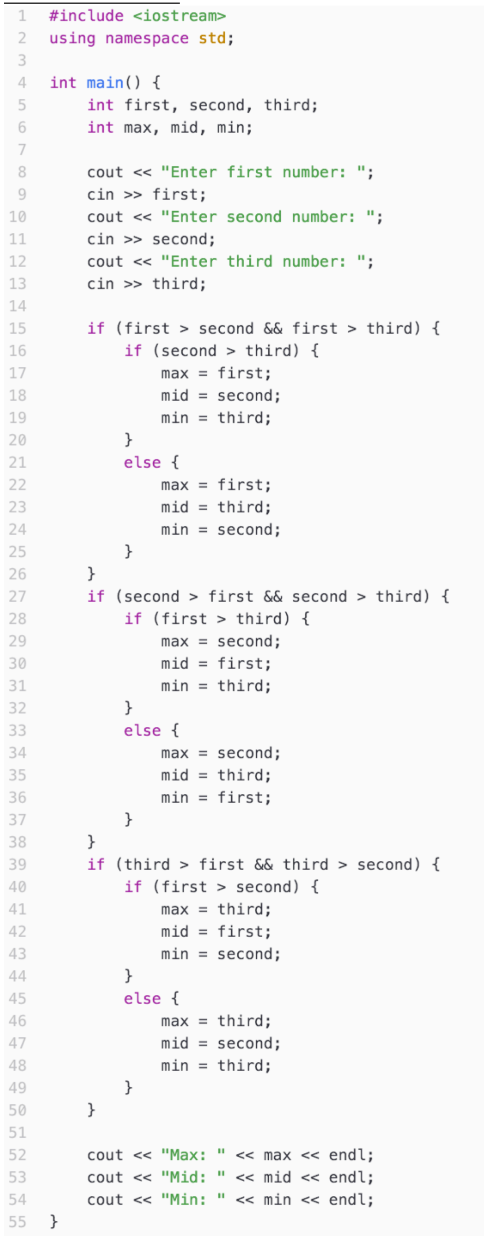
```
[thomasnguyen@Thomass-MacBook-Pro:~/Documents/Santa Clara University/Junior/COEN 171/Homework/Homework 7$ g++ five.cpp
[thomasnguyen@Thomass-MacBook-Pro:~/Documents/Santa Clara University/Junior/COEN 171/Homework/Homework 7$ ./a.out
 First all-zero row is: 4
```

Compare the readability of your code to that of the example

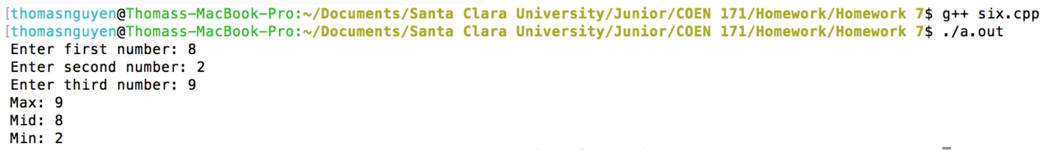
In terms of its readability, I think that my code is much more explicit as to what happens when certain logical comparisons are true. In my code, there is a flag that is raised whenever we see a nonzero character in that specific row. We are only able to identify the first all-zero row if this flag is not raised. Although the code from the textbook is much more compact and clean, it is less explicit than the code that I have written.

8.6 Consider the following programming problem: The values of three integer variables – *first, second*, and *third* – must be placed in the three variables *max, mid,* and *min*, with the obvious meanings, without using arrays or user-defined or predefined subprograms. Write two solutions to this problem, one that uses nested selections and one that does not.

First Solution

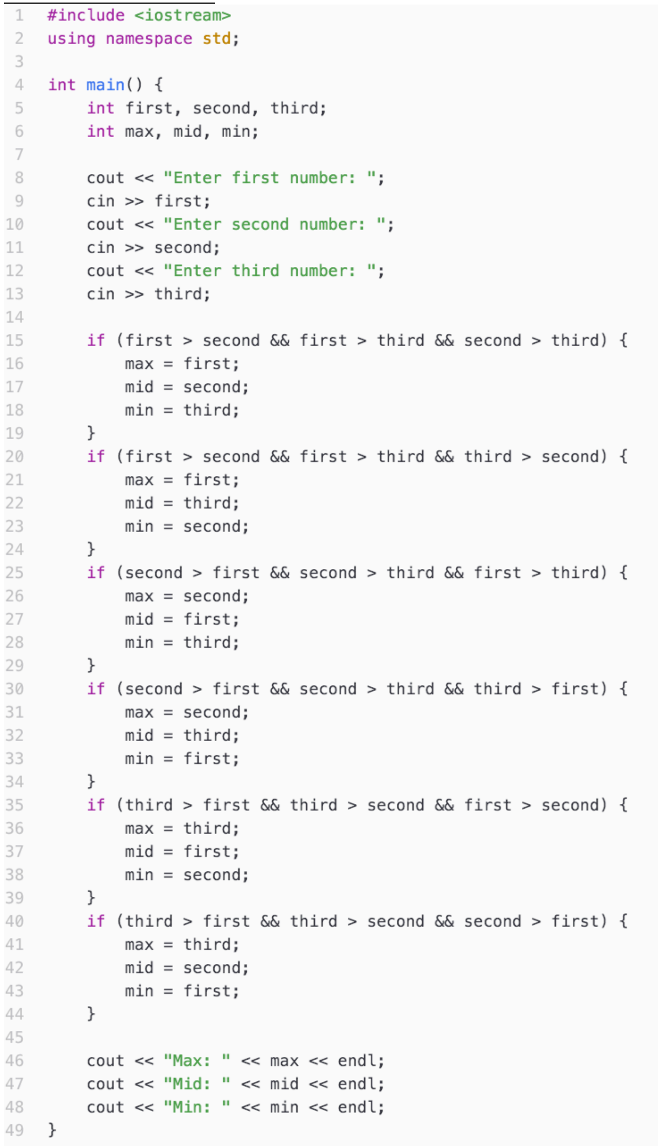


```
#include <iostream>
using namespace std;

int main() {
    int first, second, third;
    int max, mid, min;

    cout << "Enter first number: ";
    cin >> first;
    cout << "Enter second number: ";
    cin >> second;
    cout << "Enter third number: ";
    cin >> third;

    if (first > second && first > third) {
        if (second > third) {
            max = first;
            mid = second;
            min = third;
        }
        else {
            max = first;
            mid = third;
            min = second;
        }
    }
    if (second > first && second > third) {
        if (first > third) {
            max = second;
            mid = first;
            min = third;
        }
        else {
            max = second;
            mid = third;
            min = first;
        }
    }
    if (third > first && third > second) {
        if (first > second) {
            max = third;
            mid = first;
            min = second;
        }
        else {
            max = third;
            mid = second;
            min = third;
        }
    }

    cout << "Max: " << max << endl;
    cout << "Mid: " << mid << endl;
    cout << "Min: " << min << endl;
}
```

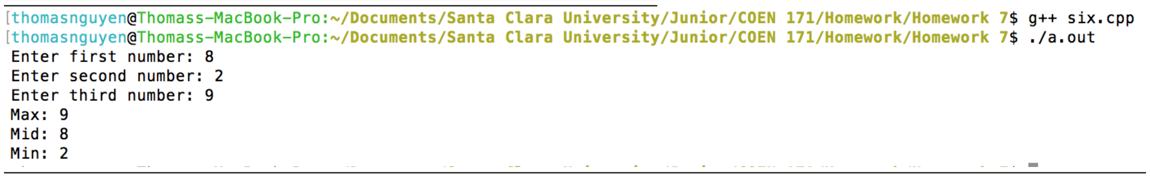


```
[thomasnguyen@Thomass-MacBook-Pro:~/Documents/Santa Clara University/Junior/COEN 171/Homework/Homework 7$ g++ six.cpp
[thomasnguyen@Thomass-MacBook-Pro:~/Documents/Santa Clara University/Junior/COEN 171/Homework/Homework 7$ ./a.out
Enter first number: 8
Enter second number: 2
Enter third number: 9
Max: 9
Mid: 8
Min: 2
```

Second Solution



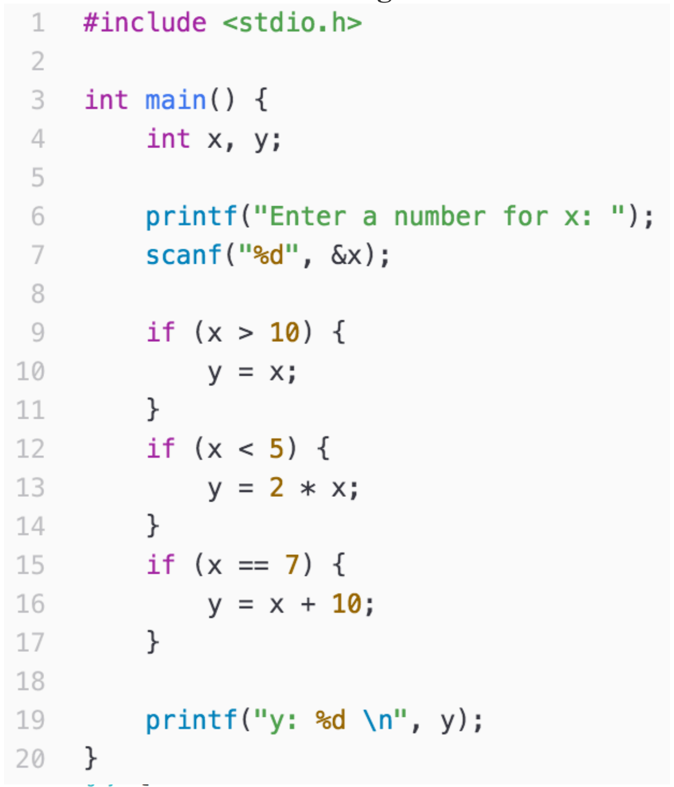
```cpp
#include <iostream>
using namespace std;

int main() {
    int first, second, third;
    int max, mid, min;

    cout << "Enter first number: ";
    cin >> first;
    cout << "Enter second number: ";
    cin >> second;
    cout << "Enter third number: ";
    cin >> third;

    if (first > second && first > third && second > third) {
        max = first;
        mid = second;
        min = third;
    }
    if (first > second && first > third && third > second) {
        max = first;
        mid = third;
        min = second;
    }
    if (second > first && second > third && first > third) {
        max = second;
        mid = first;
        min = third;
    }
    if (second > first && second > third && third > first) {
        max = second;
        mid = third;
        min = first;
    }
    if (third > first && third > second && first > second) {
        max = third;
        mid = first;
        min = second;
    }
    if (third > first && third > second && second > first) {
        max = third;
        mid = second;
        min = first;
    }

    cout << "Max: " << max << endl;
    cout << "Mid: " << mid << endl;
    cout << "Min: " << min << endl;
}
```



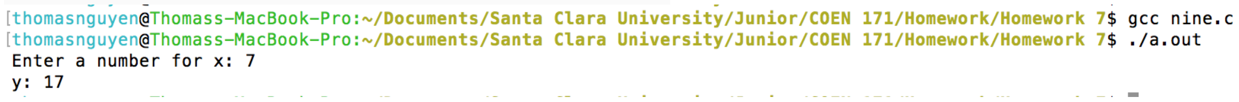
```
[thomasnguyen@Thomass-MacBook-Pro:~/Documents/Santa Clara University/Junior/COEN 171/Homework/Homework 7$ g++ six.cpp
[thomasnguyen@Thomass-MacBook-Pro:~/Documents/Santa Clara University/Junior/COEN 171/Homework/Homework 7$ ./a.out
 Enter first number: 8
 Enter second number: 2
 Enter third number: 9
 Max: 9
 Mid: 8
 Min: 2
```

Compare the complexity and expected reliability of the two.
Both are equally reliable, but the complexity varies depending on whether we nest our conditional statements or not. I find it to be less complex when we do not nest our conditional statements, but that can be subjective.

8.9 Translate the following call to Scheme's COND to C and set the resulting value to *y*



```c
#include <stdio.h>

int main() {
    int x, y;

    printf("Enter a number for x: ");
    scanf("%d", &x);

    if (x > 10) {
        y = x;
    }
    if (x < 5) {
        y = 2 * x;
    }
    if (x == 7) {
        y = x + 10;
    }

    printf("y: %d \n", y);
}
```



```
[thomasnguyen@Thomass-MacBook-Pro:~/Documents/Santa Clara University/Junior/COEN 171/Homework/Homework 7$ gcc nine.c
[thomasnguyen@Thomass-MacBook-Pro:~/Documents/Santa Clara University/Junior/COEN 171/Homework/Homework 7$ ./a.out
Enter a number for x: 7
y: 17
```