Lecture Summary

Nested **if-else** statements

It is possible to nest **if** statement within another **if** statement.

```
#include <iostream>
int main()
{
    std::cout << "Enter the values of x and y: ";
    int x{}, y{};
    std::cin >> x >> y;
    if (x > 0) { // outer if statement
        if (y > 0) { // inner if statement
            std::cout << "Both x and y are positive\n";
        }
        else {
            std::cout << "x is positive but y is not\n";
        }
    }
    else {
        std::cout << "x is not positive\n";
    }
}
```

An **if**-**else**-**if** ladder is a series of **if**-**else** statements where each **else** is followed by another **if** statement.

```
#include <iostream>
int main()
{
    int x{};
    std::cout << "Enter (x, y) coordinates: ";
    std::cin >> x >> y;
    if (x > 0 && y > 0) {
        std::cout << "Quadrant 1\n";
    }
    else if (x < 0 && y > 0) {
        std::cout << "Quadrant 2\n";
    }
    else if (x < 0 && y < 0) {
        std::cout << "Quadrant 3\n";
    }
    else if (x > 0 && y < 0) {
        std::cout << "Quadrant 4\n";
    }
    else {
        std::cout << "On an axis\n";
    }
}
```

The **if** statements are evaluated from top to bottom. The first **if** statement that evaluates to **true** is executed and the rest are skipped. To understand this better, see how the compiler translates the above code:

```
if (x > 0 && y > 0) {
    std::cout << "Quadrant 1\n";
}
else {
    if (x < 0 && y > 0) {
        std::cout << "Quadrant 2\n";
    }
    else {
        if (x < 0 && y < 0) {
            std::cout << "Quadrant 3\n";
        }
        else {
            if (x > 0 && y < 0) {
                std::cout << "Quadrant 4\n";
            }
            else {
                std::cout << "On an axis\n";
            }
        }
    }
}
```

That is, each **`else`** contains a single **`if`** statement. This is why we can skip curly braces after each **`else`** in a ladder.

Nested loops

A loop inside another loop is called a nested loop. The inner loop is executed fully for each iteration of the outer loop.

```
#include <iostream>
int main()
{
    for (int i=1; i <= 5; i++) {
        for (int j=1; j <= 5; j++) {
            std::cout << "(" << i << "," << j << ")\t";
        }
        std::cout << '\n';
    }
}
```

The fist loop will iterate from 1 to 5. For each iteration of the outer loop, the inner loop will iterate from 1 to 5. The above code will print the following output:

```
(1,1)   (1,2)   (1,3)   (1,4)   (1,5)
(2,1)   (2,2)   (2,3)   (2,4)   (2,5)
(3,1)   (3,2)   (3,3)   (3,4)   (3,5)
(4,1)   (4,2)   (4,3)   (4,4)   (4,5)
(5,1)   (5,2)   (5,3)   (5,4)   (5,5)
```

To select two different numbers from 1 to 5, we can use two nested loops as follows:

```cpp
#include <iostream>
int main()
{
    for (int i=1; i <= 5; i++) {
        for (int j=i+1; j <= 5; j++) {
            std::cout << "(" << i << "," << j << ")\t";
        }
        std::cout << '\n';
    }
}
```

The above code will print the following output:

```
(1,2)   (1,3)   (1,4)   (1,5)
(2,3)   (2,4)   (2,5)
(3,4)   (3,5)
(4,5)
```

Changing Color of Text

```cpp
#include <iostream>
using namespace std;
int main() {
cout << "\033[1;31m"; // Set text color to red
cout<<"\033[4m";
cout << "This text is red." << endl;
cout<<"Hello"<<endl;
cout << "\033[33m";
cout << "\033[0m"; // Reset text color to default
cout<<"Hello";
return 0;
}
```

Lab Questions

1. Write a program `five_per_line.cpp` that, using one `for` loop and one `if` statement, prints the integers from 1000 to 2000 with five integers per line. Only last line may have less than 5 numbers.

   Hint: use the % operator to determine when to print a newline character.

2. *Guessing game:* Generate a random number between 1 and 100. Ask the user to guess the number. Provide feedback (too high, too low) and continue until the user guesses correctly. Use a loop for repetition.

3. Write a program `star_square.cpp` that take input n and use nested `for` loops to produce the following n-by-n square pattern using these 5 color combination ($n = 5$ in the example below):

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

4. Write a program `star_triangle.cpp` that takes an input n and use nested `for` loops to produce the following output ($n = 5$ in the example below):

```
        *
      * *
    * * *
  * * * *
* * * * *
```

5. Write a program that takes an input n and use nested `for` loops to produce the following output ($n = 5$ in the example below):

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

6. Write a program that takes an input n and use nested `for` loops to produce the following output ($n = 4$ in the example below):

```
1   2   3   4
2   4   6   8
3   6   9  12
4   8  12  16
```