Lecture Summary

The `if` statement

Until last week, we have been writing programs that execute statements in sequence. However, in many cases, we need to execute a statement only if a certain condition is true. For example, we may want to print a message only if the user enters a negative number. In such cases, we use the `if` statement. The syntax of the `if` statement is as follows:

```
if (condition) {
    statement;
}
```

The `condition` is a Boolean expression that evaluates to either `true` or `false`. If the condition is `true`, the statement is executed. Otherwise, the statement is skipped. For example, the following program prints `Hello` only if the user enters a negative number.

```
#include <iostream>
int main() {
    int n;
    std::cin >> n;
    if (n < 0) {
        std::cout << "Hello\n";
    }
}
```

The `if-else` statement

Often, we want to execute one statement if the condition is true and another statement if the condition is false. For example, we may want to print `Hello` if the user enters a negative number and `Hi` otherwise. In such cases, we use the `if-else` statement. The syntax of the `if-else` statement is as follows:

```
if (condition) {
    statement1;
} else {
    statement2;
}
```

If the condition is `true`, `statement1` is executed. Otherwise, `statement2` is executed. For example, the following program prints `Negative` if the user enters a negative number and `Positive` otherwise.

```
#include <iostream>
int main() {
    int n;
    std::cin >> n;
    if (n < 0) {
        std::cout << "Negative\n";
    } else {
        std::cout << "Non-negative\n";
    }
}
```

The **if-else if- else** statement

The else if ladder allows checking multiple conditions in a sequence. If the first condition is false, the next condition is evaluated, and this continues until a true condition is found, or the default else block is executed.

```
#include <iostream>
#include <string>
using namespace std;
int main(){
    string s;
    cout<<"Enter text:"<<endl;
    getline(cin,s);
    if(s.length()>6){
        cout<<"Greater";
    }
    else if(s.length()==6){
       cout<<"Equal";
    }
    else{
        cout<<"less";
    }
}
```

The nested **if-else** statement

A nested if-else occurs when an if or else statement is placed inside another if-else. This allows more complex decision-making scenarios by combining multiple conditions.

```
#include <iostream>
using namespace std;
int main(){
    int a,b;
    cout<<"Enter two number"<<endl;
    cin>>a>>b;
    if(a%2==0 && b%2==0){
if(a>b){
    cout<<b;
}
else{
    cout<<a;
}

    }
    else if(a%2==0){
    cout<<a;
}
else if(b%2==0){
cout<<b;
}
else{
    cout<<"no even";
}
    return 0;
}
```

The **goto** statement

Goto allows you to make an absolute jump to another point in the program. You should use this feature with caution since its execution causes an unconditional jump ignoring any type of nesting limitations. The destination point is identified by a label, which is then used as an argument for the goto statement. A label is made of a valid identifier followed by a colon (:).

```
#include <iostream>
using namespace std;
int main ()
{
 int n, b;
 n=10;
b: n--;
 cout << n << ", ";
if (n==0)
 { return 0;}
goto b;}
```

Switch Statement

The switch statement objective is to check several possible constant values for an expression.

```
#include <iostream>
int main() {
    int day = 3;
    switch (day) {
        case 1:
            std::cout << "Monday" << std::endl;
            break;
        case 2:
            std::cout << "Tuesday" << std::endl;
            break;
        case 3:
            std::cout << "Wednesday" << std::endl;
            break;
        case 4:
            std::cout << "Thursday" << std::endl;
            break;

case 5:
            std::cout << "Friday" << std::endl;
            break;
        case 6:
            std::cout << "Saturday" << std::endl;
            break;
        case 7:
            std::cout << "Sunday" << std::endl;
            break;
        default:
            std::cout << "Invalid day" << std::endl;
    }
    return 0;
```

C++ common mistakes

Gotcha 1. Will the following code fragment compile? If so, what will it do?

```
int a = 10, b = 18;
if (a = b) std::cout << "equal";
else       std::cout << "not equal";
```

Solution: It uses the assignment operator `=` instead of the equality operator `==` in the conditional. This code fragment will set the variable `a` to `18` and the result of this statement is an integer, which the C++ compiler convert to **`bool`** value **`true`**. So, it prints equal without an error.

Gotcha 2. What does the following code fragment do?

```
bool a = false;
if (a = true) std::cout << "yes";
else          std::cout << "no";
```

Solution: it prints yes . Note that the conditional uses `=` instead of `==`. This means that `a` is assigned the value **`true`**. As a result, the conditional expression evaluates to **`true`**. For this reason, it is much better style to use **`if`** `(a)` or **`if`** `(!a)` when testing **`bool`** values.

Gotcha 3. What does the following code fragment do?

```
int a = 17, x = 5, y = 12;
if (x > y);
{
   a = 13;
   x = 23;
}
std::cout << a;
```

Solution: Always prints `13` since there is a spurious semicolon after the **`if`** statement. Thus, the assignment statement `a = 13;` will be executed even though `(x <= y)`. It is legal (but uncommon) to have a block that does not belong to a conditional statement, loop, or method.

Gotcha 4. What does the following code fragment do?

```
for (int x = 0; x < 100; x += 0.5) {
    std::cout << x << std::endl;
}
```

Solution: It goes into an in nite loop printing `0`. The compound assignment statement `x += 0.5` is equivalent to `x = (int)(x + 0.5)`.

Lab Questions

Q1. Write a C++ program to enter a character represent the person gender (M: for male, F: for female) and a number represents person length ((L) in inch) then find and print its perfect weight ((W) in pound) according to the following relations:

For male (M) : W = (L × 4) - 125
For female (F): W = (L × 3.5) – 108

Q2. Write a C++ program to find the roots (X1, X2) for the quadratic equation: ax2+bx+c using the formula:

$$X_1, X_2 = \frac{-b \mp \sqrt{b^2 - 4ac}}{2a}$$

Enter the constants (a, b, c) then check the following points:
1. if a= 0, display the message "Divided by zero".
2. if b2 < 4ac, display the message "No real roots".
3. if b2 = 4ac, display the message "Equal roots" then find the roots from the formula above.
4. if b2 > 4ac, display the message "real roots" then find the roots from the formula above.

Q3. Write a C++ program to find and print T from the below equations where a and b are input variables.

$$T = \begin{cases} a^2 + ab + \sqrt{ab} & \mathbf{a < b} \quad AND \quad \mathbf{a < 10} \\ ab^2 - 2a + 5b & \mathbf{a = b} \quad OR \quad \mathbf{b > 10} \end{cases}$$

Hint: For power u can use pow(double base, double exponent);

Q4. Lottery Game:
Generate a random three-digit number in the range 100-999. Take 3-digit input from user and print whether the user wins according to the following criteria:
1). If both the numbers are same then the user wins 10000 pkr.
2). If all the digits from both the numbers are same but order is different, then user wins 5000 pkr.
3). If at least 1 digit from the numbers is same the user wins 1000 pkr

Q5. Take input "n" from user and print the following using goto statement:

- Number from 1 till n
- Sum of the numbers from 1 till n

Q6. Write a program that asks the user to enter a month(1-12) and year and then displays the number of days in that month using switch case. Account for leap years for February. You are allowed to use if/else, ternary operator to check for leap years.

Hint: All years which are perfectly divisible by 4 are leap years except for century years (years ending with 00), which are leap years only if they are perfectly divisible by 400.