Lab Session 04

Multiple If-Else statements, Nested Ifs, Logical Operators and flowcharts.

Objectives:

1. To understand Multiple Nested If-Else, Ifs statement.
2. Logical Operators and flowcharts
3. Design and Compilation of C++ programs based on decision making.

Nested if...else Statements:

You're probably too young to remember adventure games on early character-mode MS-DOS systems, but let's resurrect the concept here. You moved your "character" around an imaginary landscape and discovered castles, sorcerers, treasure, and so on, using text—not pictures—for input and output.

Nested if…else statements test for multiple cases by placing if…else selection statements inside other if…else selection statements. For example, the following pseudocode if…else statement prints A for exam grades greater than or equal to 90, B for grades in the range 80 to 89, C for grades in the range 70 to 79, D for grades in the range 60 to 69 and F for all other grades:

```
If student's grade is greater than or equal to 90
cout "A"
Else
If student's grade is greater than or equal to 80
cout "B"
Else
If student's grade is greater than or equal to 70
cout "C"
Else
If student's grade is greater than or equal to 60
cout "D"
Else
cout "F"
```

This pseudocode can be written in C++ as

```
if ( studentGrade >= 90 ) // 90 and above gets "A"
cout << "A";
else
if ( studentGrade >= 80 ) // 80-89 gets "B"
cout << "B";
else
if ( studentGrade >= 70 ) // 70-79 gets "C"
```

```
cout << "C";
else
if ( studentGrade >= 60 ) // 60-69 gets "D"
cout << "D";
else // less than 60 gets "F"
cout << "F";
```

If studentGrade is greater than or equal to 90, the first four conditions are true, but only the output statement after the first test executes. Then, the program skips the else-part of the "outermost" if…else statement. Most write the preceding if…else statement as

```
if ( studentGrade >= 90 ) // 90 and above gets "A"
cout << "A";
else if ( studentGrade >= 80 ) // 80-89 gets "B"
cout << "B";
else if ( studentGrade >= 70 ) // 70-79 gets "C"
cout << "C";
else if ( studentGrade >= 60 ) // 60-69 gets "D"
cout << "D";
else // less than 60 gets "F"
cout << "F";
```

The two forms are identical except for the spacing and indentation, which the compiler ignores. The latter form is popular because it avoids deep indentation of the code to the right, which can force lines to wrap.

The following example includes a block in the else part of an if…else statement.

```
if ( studentGrade >= 60 )
cout << "Passed.\n";
else
{
cout << "Failed.\n";
cout << "You must take this course again.\n";
}
```

In this case, if studentGrade is less than 60, the program executes *both* statements in the body of the else and prints.

```
Failed.
You must take this course again.
```

Matching the else:

There's a potential problem in nested if...else statements: You can inadvertently match an else with the wrong if. BADELSE provides an example:

```
#include <iostream>
using namespace std;
int main()
{
int a, b, c;
cout << "Enter three numbers, a, b, and c:\n";
```

```
cin >> a >> b >> c;
if( a==b )
if( b==c )
cout << "a, b, and c are the same\n";
else
cout << "a and b are different\n";
return 0;
}
```

We've used multiple values with a single cin. Press Enter following each value you type in; the three values will be assigned to a, b, and c.

What happens if you enter 2, then 3, and then 3? Variable a is 2, and b is 3. They're different, so the first test expression is false, and you would expect the else to be invoked, printing *a and b are different*. But in fact nothing is printed. Why not? Because the else is matched with the wrong if. The indentation would lead you to believe that the else is matched with the first if, but in fact it goes with the second if. Here's the rule: An else is matched with the last if that doesn't have its own else.

```
Here's a corrected version:
if(a==b)
if(b==c)
cout << "a, b, and c are the same\n";
else
cout << "b and c are different\n";
```

We changed the indentation and also the phrase printed by the else body. Now if you enter 2, 3, 3, nothing will be printed. But entering 2, 2, 3 will cause the output b and c are different
If you really want to pair an else with an earlier if, you can use braces around the inner if:

```
if(a==b)
{
if(b==c)
cout << "a, b, and c are the same";
}
else
cout << "a and b are different";
```

Here the else is paired with the first if, as the indentation indicates. The braces make the if within them invisible to the following else.

while Repetition Statement:

A repetition statement specifies that a program should repeat an action while some condition remains true. The pseudocode statement

```
While there are more items on my shopping list
Purchase next item and cross it off my list
```

Describes the repetition that occurs during a shopping trip. The condition, "there are more items on my shopping list" is either true or false. If it's true, then the action, "Purchase next item and cross it off my list" is performed. This action will be performed repeatedly while the condition

remains true. The statement contained in the *While* repetition statement constitutes the body of the *While*, which can be a single statement or a block. Eventually, the condition will become false (when the last item on the shopping list has been purchased and crossed off the list). At this point, the repetition terminates, and the first pseudocode statement after the repetition statement executes.

As an example of C++'s while repetition statement, consider a program segment designed to find the first power of 3 larger than 100. Suppose the integer variable product has been initialized to 3. When the following while repetition statement finishes executing, product contains the result:

```
int product = 3;
while ( product <= 100 )
product = 3 * product;
```

When the while statement begins execution, product's value is 3. Each repetition multiplies product by 3, so product takes on the values 9, 27, 81 and 243 successively. When product becomes 243, the while statement condition—product <= 100—becomes false. This terminates the repetition, so the final value of product is 243. At this point, program execution continues with the next statement after the while statement.

Examination-results problem: Nested control statements:

```
#include <iostream>
using namespace std;
int main()
{
// initializing variables in declarations
int passes = 0; // number of passes
int failures = 0; // number of failures
int studentCounter = 1; // student counter
int result; // one exam result (1 = pass, 2 = fail)

// process 10 students using counter-controlled loop
while ( studentCounter <= 10 )
        {
        // prompt user for input and obtain value from user
        cout << "Enter result (1 = pass, 2 = fail): ";
        cin >> result; // input result
        // if...else nested in while
        if ( result == 1 ) // if result is 1,
                passes = passes + 1; // increment passes;
        else // else result is not 1, so
                failures = failures + 1; // increment failures
        // increment studentCounter so loop eventually terminates
        studentCounter = studentCounter + 1;
        } // end while
        // termination phase; display number of passes and failures
        cout << "Passed " << passes << "\nFailed " << failures << endl;
        // determine whether more than eight students passed
        if ( passes > 8 )
        cout << "Bonus to instructor!" << endl;
```

```
} // end main
```

State the O/P of above program:

Else-if Instruction:

Assignment:

1. **What is Preincrementing and postincrementing, explain with the help of c++ program. (Reference Material: C++ Deital Deital, Chapter # 04 - 4.12)**
2. **State pre-fix and post-fix. (Reference Material: C++ Deital Deital, Chapter # 04 - 4.12)**
3. **Write C++ statements to accomplish each of the following tasks.**
   - **a) Declare variables sum and x to be of type int.**
   - **b) Set variable x to 1.**
   - **c) Set variable sum to 0.**
   - **d) Add variable x to variable sum and assign the result to variable sum.**
   - **e) Print "The sum is: " followed by the value of variable sum.**

4. **Write single C++ statements or portions of statements that do the following:**
   - **a) Input integer variable x with cin and >>.**
   - **b) Input integer variable y with cin and >>.**
   - **c) Set integer variable i to 1.**
   - **d) Set integer variable power to 1.**
   - **e) Multiply variable power by x and assign the result to power.**