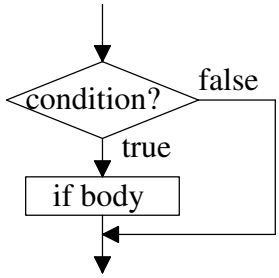
Name:_____

1. An `if` statement allows code to be executed or not based on the result of a comparison. The syntax of an `if` statement is:

```
if (condition) {
    statement1;
    statement2;
    statement3;
} // end if
```



If the condition evaluates to `true`, then the statements of the indented body is executed. If the condition is `false`, then the body is skipped. The statements in the body of the `if` statement are indented.

Typically, the condition involves comparing "stuff" using relational operators (`<, >, ==, <=, >=, !=`). For example, we might want to print "Your grade is A." if the variable `score` is greater-than or equal to 90.

```
if (score >= 90) {
    cout << "Your grade is A.";
} // end if
```

Complex conditions might involve several comparisons combined using Boolean operators: `!(not), || (or), && (and)`. For example, we might want to print "Your grade is B." if the variable `score` is less than 90, but greater than or equal to 80.

```
if (score < 90 && score >= 80) {
    print "Your grade is B.";
} // end if
```

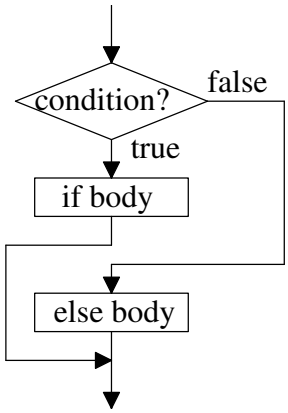
Using `if` statements, write C++ code to output the appropriate string according to variable `temperature`'s value.

Temperature	**String**
temperature < 0	"Its bitterly cold!"
0 <= temperature <= 32	"Its freezing outside."
32 < temperature < 68	"Light jacket weather."
68 <= temperature	"Its warm outside."

Name:_____

2. An `if/else` statement allows a block of code to be executed if the result of a comparison is `true`; otherwise the "else" block of code will be executed. The syntax of an `if/else` statement is:

```
if (condition) {
    // only executed if the condition is True
    statement_T1;
    statement_T2;
    statement_T3;
} else {
    // only executed if the condition is False
    statement_F1;
    statement_F2;
    statement_F3;
}
```



Using only nested `if/else` statements, write C++ code to output the appropriate string according to variable temperature's value.

Temperature	**String**
temperature < 0	"Its bitterly cold!"
0 <= temperature <= 32	"Its freezing outside."
32 < temperature < 68	"Light jacket weather."
68 <= temperature	"Its warm outside."

Name:_____

3. Using `if/else if` format, write C++ code to output the appropriate string according to variable temperature's value.

Temperature	**String**
temperature < 0	"Its bitterly cold!"
0 <= temperature <= 32	"Its freezing outside."
32 < temperature < 68	"Light jacket weather."
68 <= temperature	"Its warm outside."

4. Draw a flow-chart for the above `if/else if` formatted code.

5. Correct the syntax and logic errors in the following code:

```
if  x < 0  {
     cout << "x = " << x << endl;
     cout << "x is negative." << endl;
} else if (x = 0)
     cout << "x is zero." << endl;
} else if (0 <= x < 10) {
     cout << "x = " << x << endl;
     cout << "x is a small positive." << endl;
} else
     cout << "x = " << x << endl;
     cout << "x is a large positive." << endl;

cout << "Bye!" << endl;
```

Name:_____

Operator Precedence and Associativity		
Operator	**Associativity**	**Usage(s)**
`::`	unary: left-to-right binary: right-to-left	
`() [] -> .`	left-to-right	parenthesis index object pointer/structure pointer dot operator
`++ -- - + ! ~ (type) * & sizeof`	right-to-left	increment and decrement unary negation and plus logical negation one's complement operator type cast indirection address-of/reference
`* / %`	left-to-right	multiply, division, remainder
`+ -`	left-to-right	addition and subtraction
`<< >>`	left-to-right	io: insertion and extraction bit-wise shift left and right
`< <= > >=`	left-to-right	comparisons for inequality
`== !=`	left-to-right	comparison for equality
`&`	left-to-right	bit-wise AND
`^`	left-to-right	bit-wise exclusive-OR
`|`	left-to-right	bit-wise OR
`&&`	left-to-right	logical AND
`||`	left-to-right	logical OR
`?:`	right-to-left	conditional
`= += -= *= /= %= &= ^= |= <<= >>=`	right-to-left	assignment
`,`	left-to-right	comma operator