# Level 2: Nested Conditions

*Exercise 1*: Every morning at 10 am, Lucy looks at the thermometer and the humidity meter and decides what to eat or drink according to the temperature and the humidity levels. The following table shows what Lucy eats or drinks in what conditions. For example, if the temperature is 45°F and the humidity is 53% then Lucy eats cake.

| | **Temperature < 60°F** | **Temperature ≥ 60°F** |
|---|---|---|
| **Humidity < 40%** | Lucy drinks coffee | Lucy drinks iced tea |
| **Humidity ≥ 40%** | Lucy eats cake | Lucy eats ice cream |

Write a program that reads two numbers, the temperature and the humidity accordingly, from the input file and writes what Lucy does to the output file.

*Sample*:

| *inFile.txt*: | *outFile.txt*: |
|---|---|
| `45 53` | `Lucy eats cake` |

*Solution*:

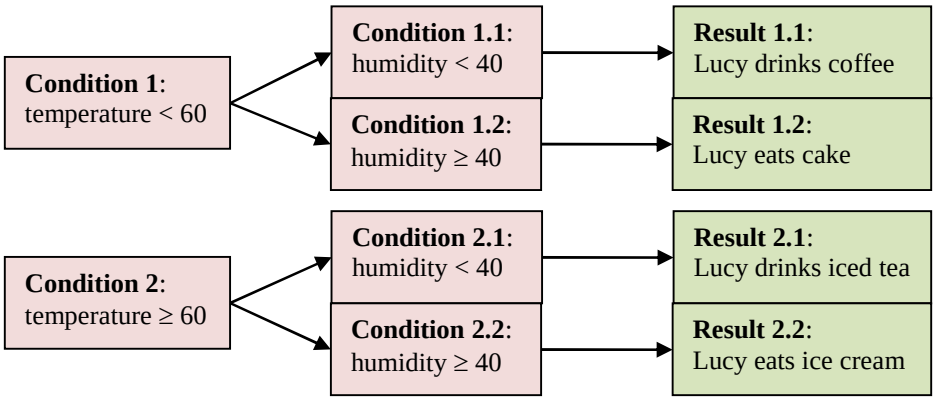
```
#include <fstream>
using namespace std;

int main ()
{
  int temperature, humidity;
  ifstream fin ("inFile.txt");
  ofstream fout ("outFile.txt");

  fin >> temperature >> humidity;
  if (temperature < 60 && humidity < 40)
    fout << "Lucy drinks coffee" << endl;
  if (temperature < 60 && humidity >= 40)
    fout << "Lucy eats cake" << endl;
  if (temperature >= 60 && humidity < 40)
    fout << "Lucy drinks iced tea" << endl;
  if (temperature >= 60 && humidity >= 40)
    fout << "Lucy eats ice cream" << endl;
}
```

The first two `if`s have a shared condition which the temperature is less than 60°F. Also, the last two `if`s have another shared condition. We can put `if`s into two groups by their first conditions as in the figure below. If we check the first condition we reduce the

conditions to two out of four. And then we can check the other condition separately and decide which action Lucy will take.




The following code shows how we can compare two conditions one after the other. This is called "nested `if`s". In the first '`if` block', we only consider the conditions that satisfies '`temperature < 60`'.

```
if (temperature < 60)
{
  if (humidity < 40)
    fout << "Lucy drinks coffee" << endl;
  else
    fout << "Lucy eats cake" << endl;
}
```

Not that the `else` condition corresponds to '`temperature < 60 && humidity >= 40`'.

**Note**: In the code above, we may not use parenthesis since the inner `if` block is only one statement.

*Exercise 2*: Write the same program in the previous exercise by using nested `if`s.

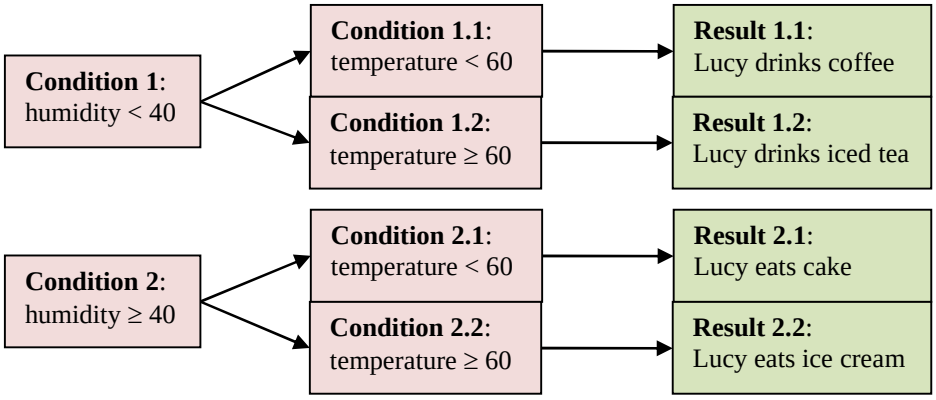
*Solution*:

```
1.   #include <fstream>
2.   using namespace std;
3.
4.   int main ()
5.   {
6.     int temperature, humidity;
7.     ifstream fin ("inFile.txt");
8.     ofstream fout ("outFile.txt");
9.
10.    fin >> temperature >> humidity;
11.    if (temperature < 60)
12.      if (humidity < 40)
13.        fout << "Lucy drinks coffee" << endl;
14.      else
```

```
15.        fout << "Lucy eats cake" << endl;
16.    else
17.      if (humidity < 40)
18.        fout << "Lucy drinks iced tea" << endl;
19.      else
20.        fout << "Lucy eats ice cream" << endl;
21.  }
```

The block in the lines 12-15 belongs to the `if` condition '`temperature < 60`'. The `else` in line 16 corresponds to the condition '`temperature >= 60`', and the block in lines 17-20 belongs to it.

In the last exercise, we grouped the conditions into two groups by their temperatures. Similarly, if we look at the humidity values we observe that two of them share one condition while the other two share another condition. So, we can group them by their humidity levels as well. Another way of using nested `if`s is to check their humidity levels first, and then the temperatures to decide. This approach is just the swapping the conditions in the previous one. The following chart summarizes the approach:




*Exercise 3*: Write the same program in the previous exercise checking the humidity and then checking the temperature in nested `if`s.

*Exercise 4*: Write the same program with the conditions below <u>without</u> using nested `if`s.

| | **Temperature < 60°F** | **Temperature ≥ 60°F** |
|---|---|---|
| **Humidity < 40%** | Lucy drinks coffee | Lucy drinks iced tea |
| **40% ≤ Humidity < 60%** | Lucy plays tennis | Lucy plays golf |
| **Humidity ≥ 60%** | Lucy eats cake | Lucy eats ice cream |

*Exercise 5*: Write the same program using nested `if`s.

*Exercise 6*: Consider the same example, but this time you will have `temperature` and `humidity` information for more days. You will output the days Lucy plays tennis or golf. In the input file, the first line will provide `N`, the number of days. Upcoming N lines will provide the `temperature` and `humidity` information for these days.

*Sample*:

| *inFile.txt*: | *outFile.txt*: |
|---|---|
| `4`<br>`45 83`<br>`58 58`<br>`77 32`<br>`77 40` | `Day 2: Lucy plays tennis`<br>`Day 4: Lucy plays golf` |

*Exercise 7*: Lucy wants to calculate her bonus points she earned while shopping. She knows how much she spent in her last N ≤ 100 shopping, and all of them are *bakery*, *produce*, or *drinks*.

- If they are *bakery*, she earns 5 points per dollar.
- If they are *produce*, she earns 2 points per dollar.
- If they are *drinks*, she earns 1 point per dollar.

She earns bonus points when she spends more than M dollars in one shopping. Input file is '*shopping.txt*'. You will first read N then N numbers corresponding to the money Lucy spend. At the end, you will read M then the type of shopping items. 1 means they are all *bakery*; 2 and 3 correspond to *produce* and *drinks* respectively. Write the total bonus points to the output file '*bonus.txt*'.

*Sample*:

| *shopping.txt*: | *bonus.txt*: |
|---|---|
| `5`<br>`41 13 88 50 72`<br>`50 1` | `800` |

Lucy receives bonus points for shopping more than $50 hence $88 and $72 are qualified for bonus. Her shopping was on *bakery* which brings 5 points per dollar. In total, she got (88 + 72) * 5 = 800 bonus points.

*Exercise 8*: Sort given tree numbers from smaller to larger without using loops and nested `ifs`.

*Sample*:

| *numbers.txt*: | *sorted.txt*: |
|---|---|
| `41 13 27` | `13 27 41` |

* *Exercise 9*: Sort given tree numbers from smaller to larger using nested `ifs` but not loops.