

## COSC 2P95 – Lab Exercise 6 – Object Orientation

### Background:

Have you ever noticed how difficult it is to organize ordering pizza for a large group?

- Just to be clear: the answer is ‘yes’. Doesn’t matter if it makes sense; we’re going with ‘yes’

Shelly likes a classic pepperoni and cheese on red sauce. Marshall likes olives and artichokes on white. Gary just likes red and white sauce with nothing else on. We... don’t invite Marshall to our pizza parties anymore.

The more people you’re ordering for, the more you have to compromise. e.g. if Allen wants a classic Hawaiian (ham, pineapple, and cherries), and Brenda wants a more local variant (bacon and pineapple), then clearly the solution is to take the intersection of the two: nothing but pineapple!

(Yes yes, probably sauce and cheese also, but the point is: pineapple is great and everyone loves it)

### Pizza:

To assist with this, we could devise a new data type: a **Pizza**.

To be clear, a `Pizza` in this context isn’t the pizza itself, but rather a *configuration of toppings*.

- There are 20 possible toppings (including sauces and cheeses)
  - The options are all defined in an *unscoped enumeration*
  - Each topping is either present, or not (e.g. you can’t get ‘double bacon’)
- Since a `Pizza` is really just ‘one possible configuration’, the type is *immutable*. If you want a different configuration, simply use one of the included operators to create new derived configuration
  - e.g. if `p1` is `[Red sauce / Pepperoni]`, then `p2=p1+OLIVES`; means `p2` ends up with `[Red sauce / Pepperoni / Olives]`, but `p1` remains unchanged

### Pizza Operators:

Operator	C++ [returns]
Union of toppings present in <i>either</i> pizza	<code>A+B</code> [ <code>Pizza</code> ]
Intersection of only toppings common to <i>both</i> pizzas	<code>A^B</code> [ <code>Pizza</code> ]
Result of removing toppings defined by a different pizza	<code>A-B</code> [ <code>Pizza</code> ]
Result of adding a single topping to the pizza	<code>A+b</code> [ <code>Pizza</code> ]
Result of leaving a single topping off the pizza	<code>A-b</code> [ <code>Pizza</code> ]
Checking if a pizza’s toppings are contained entirely within another’s	<code>A&lt;=B</code> [ <code>bool</code> ]
Checking if second pizza’s topping are a strict superset of this one’s	<code>A&lt;B</code> [ <code>bool</code> ]
Checking if a pizza has <i>at least</i> all the same toppings as another	<code>A&gt;=B</code> [ <code>bool</code> ]
Checking if a pizza has the same toppings as another, plus at least one more	<code>A&gt;B</code> [ <code>bool</code> ]
Checking if two pizzas have identical toppings	<code>A==B</code> [ <code>bool</code> ]
Checking if two pizzas are different	<code>A!=B</code> [ <code>bool</code> ]
Checking if a pizza has <i>no</i> toppings at all	<code>!A</code> [ <code>bool</code> ]
Getting the <i>number</i> of toppings on the pizza	<code>A()</code> [ <code>int</code> ]
Checking if a <i>specific</i> topping is present	<code>A[t]</code> [ <code>bool</code> ]
Stream insertion (output)	<code>O&lt;&lt;A</code> [ <code>ostream</code> ]

There are also two **static** functions for presets: one that yields the configuration of *all* possible toppings, and one that’s just for solely the *meats* (i.e. not even sauce or cheese).

### Task:

You’ve been provided with a header file. **Do not modify it.**

You’ll be writing two files:

- `Pizza.cpp` – containing the complete implementation, relying on `Pizza.h`
- A simple program to demonstrate all of the operations

The output (stream insertion) uses the following format:

- Starts with [ and ends with ]
- Each topping is listed in plain English
  - Each topping is separated by a |
  - There is no | before the first topping, or after the last one
  - An 'empty' pizza has no | at all

e.g.

[Pepperoni]

[]

[Red sauce | White sauce | Mozzarella | Feta | Pepperoni | Green pepper | Red pepper | Banana pepper | Black pepper | Sausage | Bacon | Pineapple | Corn | Artichoke | Toes | Mushrooms | Onions | Ham | Olives | Cherries]

If you're still having issues by then, we'll cover a couple tips during lecture.

### **Submission:**

In addition to your source files, also include a sample execution or two.

Reminder: if you need to bundle up files, .zip is the only permitted format!