# HW8

**Reading: Chapter 8**

**Programming**:

Submit a single file named hw8.py that contains the solutions to the two problems below. When you are finished, test your solutions using `doctest`. Include the following code at the bottom of your module in order to run the doctest:

```
if __name__=='__main__':
    import doctest
    print( doctest.testfile( 'hw8TEST.py'))
```

1. (Composition) Write a `Pizza` class so that this client code works. Please note that it is ok if the toppings are listed in a different order.

```
>>> pie = Pizza()
>>> pie
Pizza('M',set())
>>> pie.setSize('L')
>>> pie.getSize()
'L'
>>> pie.addTopping('pepperoni')
>>> pie.addTopping('anchovies')
>>> pie.addTopping('mushrooms')
>>> pie
Pizza('L',{'anchovies', 'mushrooms', 'pepperoni'})
>>> pie.addTopping('pepperoni')
>>> pie
Pizza('L',{'anchovies', 'mushrooms', 'pepperoni'})
>>> pie.removeTopping('anchovies')
>>> pie
Pizza('L',{'mushrooms', 'pepperoni'})
>>> pie.price()
16.65
>>> pie2 = Pizza('L',{'mushrooms','pepperoni'})
>>> pie2
Pizza('L',{'mushrooms', 'pepperoni'})
>>> pie==pie2
True
```

The `Pizza` class should have two attributes(data items):

`size` – a single character str, one of 'S','M',L"

`toppings` – a `set` containing the toppings. If you don't remember how to use a `set`, make sure you look it up in the book. Please note that toppings may be listed in a different order, but hw2TEST.py takes that into account.

The `Pizza` class should have the following methods/operators):

`__init__` - constructs a Pizza of a given size (defaults to 'M') and with a given set of toppings (defaults to empty set). I highly recommend you look at the Queue class in the book to see how to get this to work correctly.

`setSize` – set pizza size to one of 'S','M'or 'L'

`getSize` – returns size

`addTopping` – adds a topping to the pizza, no duplicates, i.e., adding 'pepperoni' twice only adds it once

`removeTopping` – removes a topping from the pizza

`price` – returns the price of the pizza according to the following scheme:

'S': $6.25 plus 70 cents per topping
'M': $9.95 plus $1.45 per topping
'L': $12.95 plus $1.85 per topping

`__repr__` - returns representation as a string – see output sample above. Note that toppings may be listed in a different order.

`__eq__` - two pizzas are equal if they have the same size and same toppings (toppings don't need to be in the same order)

2. Write a function `orderPizza` that allows the user input to build a pizza. It then prints a thank you message, the cost of the pizza and then **returns** the Pizza that was built.

```
>>> orderPizza()
Welcome to Python Pizza!
What size pizza would you like (S,M,L): M
Type topping to add (or Enter to quit): mushroom
Type topping to add (or Enter to quit): onion
Type topping to add (or Enter to quit): garlic
Type topping to add (or Enter to quit):
Thanks for ordering!
Your pizza costs $14.299999999999999
Pizza('M',{'mushroom', 'onion', 'garlic'})
>>> orderPizza()
Welcome to Python Pizza!
What size pizza would you like (S,M,L): L
Type topping to add (or Enter to quit): calamari
Type topping to add (or Enter to quit): garlic
Type topping to add (or Enter to quit):
Thanks for ordering!
Your pizza costs $16.65
Pizza('L',{'garlic', 'calamari'})
>>> p=orderPizza()
Welcome to Python Pizza!
What size pizza would you like (S,M,L): S
Type topping to add (or Enter to quit):
```

```
Thanks for ordering!
Your pizza costs $6.25
>>> p
Pizza('S',set())
>>>
```