Problem 1: Basics

Checking to see if an arbitrary string of size 5 contains balanced parentheses can be done via a regular expression

- True True

Checking to see if an arbitrary string contains balanced parenthesis can be done via a regular expression

- True True

Languages that support higher order programming allow you to return functions from other functions

- True True

Python uses a Dynamic Type System

- True True

Python uses a Static Type System

- True True

If a language uses a static type system, it means it also uses an explicit type system

- True True

If a language uses an explicit type system, it means it also uses an static type system

- True True

Problem 2: Python Higher Order Programming

Python has a built in function called filter(). It takes in a list and a function and will return a list of all values that passed the filter from the input list. Examples:

- list(filter(lambda x: x > 4, [1,2,3,4,5,6])) == [5,6]
- list(filter(lambda x: (x + 1)%3 == 0, [1,2,3,4,5,6])) == [2,5]

Write your own filter method using reduce. You may not use any looping structure. Hint: a nested function OR a lambda might be the way to go.

```python
def my_filter(f, lst):
```
**Problem 3: Regex in Python**

(a) Which of the following strings are an exact match of the following Regular Expression? Mark all that apply. [2 pts]

```
ˆ([A-Z][a-z])+([0-9]+)?([A-Za-z])?$
```

- A AbCd123Ef
- B AbCd
- C AbCD
- D AbC
- E AbCd123E
- F ABC
- G None

(b) [2 pts]

Consider the following strings:

"Name: Cliff" "Name: Kauffman" "Major: PHIL" "Major: CS"

Which regex would accept all the above strings? It is okay if they accept other strings as well. Mark all that apply.

- A `^[A-Z][a-z]+: [A-Za-z]+$`
- B `(Name|Major): (Cliff|Kauffman|PHIL|CS)`
- C `[A-z]+:.([A-Z]+|[A-Z][a-z]+)`
- D None

**Problem 4: Putting it all together**

[Total 8 pts]

Using either map or reduce, and given a list of phone numbers, implement `get_area` and `sum_area` to return the sum of the area codes. You may not use any looping structure (for, while, etc), nor can you use `.split()`. You write as many regexes as you think you will need.

Valid Phone numbers consist of 10 (ten) digits and will take one of the following formats. If a phone number is incorrectly formatted, ignore it.

**Phone Formats:**
- XXXXXXXXXX
- (XXX)XXXXXXX
- (XXX)-XXX-XXXX

**Helpful Regex Things:**
- `re.match(regex,string)` returns a match object or None if not matched
- `matched.group(x)` returns the substring captured by group x

**Some shortcuts:**
- `+`: one or more repetitions
- `?:` Zero or one repetitions
- `[^x]`: Anything but x

**For example:**

numbers = ["1234567890","(111)-222-3333","(000)2223333","(invalid)"]

sum_area(numbers) = 234  # 123 + 111 + 000

def get_area(phone_number):
    # return the area code of a phone number


def sum_area(numbers):
    areas = map(get_area,numbers)
    total = reduce( ... , areas, 0)  # fill in the blank
    return total