Name: _____ Recitation: _____ Andrew Id: _____

**15-112 Spring 2018 Quiz 8**

Up to 20 minutes. No calculators, notes, books, or computers. Do not use recursion. Show your work!

1. (10 points) **Short Answer** Give an example of a value that cannot be added to a set, and explain why it can't be added.

2. (30 points) **Code Writing:** Write the function mostAppearances(chapters). This function should take a dictionary that maps chapters to lists of characters who appear in the chapter, and should return a set of the characters that appear most often across all the chapters. For example, given the dictionary:

```
{ "Third" : [ "Ender", "Peter", "Val", "Stilson" ],
  "The Giant's Drink" : [ "Graff", "Ender", "Bernard", "Alai" ],
  "Locke and Demosthenes" : [ "Graff", "Peter", "Val", "Ender", "Petra" ],
  "Valentine" : [ "Val", "Ender", "Graff" ] }
```

The function should return `{ "Ender" }`, since "Ender" appears in all four chapters. If "Ender" did not appear in the first list, then it would return `{ "Ender", "Val", "Graff" }`, since each name would occur in three of the four lists.

3. (15 points) **Code Writing:** The piece of code shown below will run in $O(N^2)$ time. In the space under the code, write a new version of the function that performs the same operation but runs in $O(N)$ time instead.

```
def reverseNums(lst):
    newLst = []
    for i in range(len(lst)):
        if isinstance(lst[i], int):
            newLst.insert(0, lst[i])
    return newLst
```

| Built-in Big-O Runtimes | |
|---|---|
| General | |
| `isinstance(item, type)` | $O(1)$ |
| `len(item)` | $O(1)$ |
| `item[i]` | $O(1)$ |
| Strings | |
| `c in s` | $O(N)$ |
| Lists | |
| `lst.append(item)` | $O(1)$ |
| `lst[i:j:k]` | $O((j-i)/k)$ |
| `lst.insert(i, item)` | $O(N)$ |
| `item in lst` | $O(N)$ |
| `min(lst) / max(lst)` | $O(N)$ |
| `lst.reverse()` | $O(N)$ |
| `lst.sort()` | $O(NlogN)$ |

4. (45 points) **Short Answer:** For each of the three functions shown below, write next to each line of the function either the Big-O runtime of the line or the number of times the line loops. Then write the total Big-O runtime of the function in terms of N in the box to the right of the code.

```
1: # lst1 & lst2 are lists of length N
2: def sa1(lst1, lst2):                              # Big-O
3:     x = 0                                         # _____
4:     for i in range(len(lst1)):                    # _____
5:         if lst1[i] in lst2:                       # _____
6:             for j in range(len(lst2)-1, -1, -1):  # _____
7:                 if lst1[i] == lst2[j]:            # _____
8:                     x += 1                        # _____
9:     return x                                      # _____
```

```
1: def sa2(lst): # lst is a list of length N         # Big-O
2:     if len(lst) == 0:                             # _____
3:         return False                              # _____
4:     if lst[0] != min(lst):                        # _____
5:         lst.sort()                                # _____
6:     tmp = lst[::2]                                # _____
7:     return max(lst) in tmp                        # _____
```

```
1: def sa3(s): # s is a string with N characters     # Big-O
2:     for letter in string.ascii_uppercase:         # _____
3:         if s[-1] == letter:                       # _____
4:             return ""                             # _____
5:     i = len(s) - 1                                # _____
6:     result = ""                                   # _____
7:     while i >= 0:                                 # _____
8:         result += s[int(i)]                       # _____
9:         i -= len(s) / 4                           # _____
10:    return result                                 # _____
```