Name: AndrewID:

**15-112 Exam #2**

**Summer ‘13/Kesden**

1. **Write a function** that accepts arguments x and y, and uses a **while** loop to return the remainder when the number *x* is divided by the number *y*. Please note that *x* and *y*, can each be positive, negative, or zero, and are not given in any particular order, e.g. *x* is **not** necessarily less than *y*. You may not use the /-divide operator.

2. Given integer variables *x* and *y*, if both *x* and *y* are non-positive, use a single conditional to **raise an exception** (any type of exception will do).

3. Define a *NegativeAmountError* Exception.

4. Assume that the function creditChecking(amount) is intended to add money to some checking account, not subtract it. Further assume that it raises an *NegativeAmountError* exception should a negative amount be passed in. Write a code segment that calls this function, passing in a value via the variable *unvalidatedAmount*, and, should this exception be thrown, prints "Amount must be non-negative."

5. The members of your course staff are "Daniel", "TJ", and "Aaron". Write code that adds these three folks to a **Set**, and then uses the set to determine if "Dave" is a member of the course staff, printing "Dave is a member" or "Dave is not a member", as appropriate.

6. Given two sets, *instructors* and *TAs,* create a third set, *courseStaff*, that contains the union (all members in either) of the other two sets.

7. Assume that you are given a **List** called, *courseStaff*. Use the loop of your choice to print the items within the list in reverse order.

8. Write a function, recordReservation(reservations, time, name). This function should accept a **Dictionary**, *reservations*, and strings, *time* and *name*. It should add the mapping *time* (key)→*name* (value) to *reservations*.

9. Write a complimentary function findReservation(reservations, time) that **returns** (**not** prints) the name of the person holding a reservation at a particular time.

10. Write another complimentary function findDuplicateReservations(reservations) that **prints** (**not** returns) the name of any person with more than reservation, as well as the times of the reservations. You may format the output however you'd like.

11. Write a **recursive** function, sumValues(numbers), which uses recursion to return the sum of the integer values contained within the **List** passed in as numbers.

12. Please write a **generator**, *pow2()* that produces non-fractional powers of 2, e.g 1, 2, 4, 8, 16, 32.

13. Please **initialize** and otherwise **use your generator** from question #1 to print out the first 3 powers of 2.

14. Please write a **co-routine**, *runningTotal()*. When initialized, its total should be zero. Each time it is given an integer number, it should add that number to a running total and return the running total.

15. Please initialize and call an instance of the *runningTotal()* co-routine described above, use it to add the numbers 4, 5, and 6, and print the running total after each number is added.