COMP201 — Principles of Programming

Semester I 2009–2010

Quiz #9

Student ID: _____ **Name:** _____ **Score:** _____

IMPORTANT: This programming environment is not the same as the one that you are using in class. To save and compile your programs, follow the instructions below:

- Log onto the machine with the following information:
  - Location: ShortCourse
  - Username: `s` + *student id*. For example, if your student ID is 09123456d, log in with the user name `s09123456d`. All lowercase letters.
  - Password: comp201
- Run `nalwin32` and get to Python just like you do in class.
- JEdit 4.2 should be available directly from the **Start** menu, or on your desktop, or from `nalwin32`.
- **Save your programs onto the `J:` drive. Do not create any subfolders.** We will be using a program to collect your programs, and if you don't save it in the correct place, we won't be able to get it and you will get no marks.

1. A mathematical *set* is a collection of data with the following characteristics:

   - There is no ordering between the elements in a set.
   - An element is either in the set, or not in the set. Therefore, a set does not contain repeated elements.

   For example:

   - { 3, 5, 2, 6, 9 } is a valid set.
   - This set is equal to the set { 5, 6, 2, 9, 3 }, because they have the same elements (remember that the order doesn't matter in a set).
   - Something like { 3, 5, 2, 6, 2, 9 } would not be a valid set (a set cannot have repeated elements).

   For this question, we want you to use a Python list to make a set. In other words, we want you to store all the data items into a list, but to ensure that the set properties apply to that list. Your set will be used for storing integers.

   The only list operations that you may use for this quiz are:

   - indexing and slicing
   - concatenation and repetition (`+` and `*`)
   - iteration through the list (for loops)
   - `len()`
   - `del()`
   - `append()`
   - `pop()`

   There are three data files in your `J:` drive: `setData.txt`, `setData1.txt` and `setData2.txt`. They contain randomly-generated integers (some of them may be repeated). This will be the data that you need to store into your set.

   As a hint, the following snippet of code will read in all lines from the file `test.txt`:

```
infile = open("test.txt", "r")
for line in infile:
    # do something with line
infile.close()
```

   There will be multiple levels to this question. Programs for different levels will have different names. In the case that a student has multiple programs in his/her `J:` drive, we will grade the one **with the highest level**. Therefore, please delete any unnecessary files.

Level 1: Students finishing this level will be guaranteed to get **at least a pass** for their continuous assessment grade.

For this level, you need to:

- Write a function, `addToSet()`, that will take as parameters, a reference to a set, and a data item. It should then add that data item to the set.
- Write a function, `isInSet()`, that will take as parameters, a reference to a set, and a data item. It should return `True` if the data item is in the set, and `False` otherwise.
- Write a function, `makeSet()`, that will take the name of a file as a parameter. It should read in data items from that file and create a set from it. It should return the set.
- Write a function, `sizeOfSet()`, that will take as parameter a reference to a set, and return the size (number of elements) in the set.
- Write a function, `deleteFromSet()`, that will take as parameters, a reference to a set, and an element (which may or may not be in the set.) If the element exists in the set it should delete that element from the set.
- Write a function, `printSet()`, that will print out the contents of the set. In mathematics, a set is enclosed in curly brackets ({..}).

The `main()` function for your program is shown below, and can be found in your `J:` drive as `Level1Main.py`. You are **NOT** allowed to change anything in `main()`.

```
def main():
    print("This Level 1 program makes a set")
    set1 = readInData("setData.txt")
    print("Set 1:", end = "")
    printSet(set1)
    print("Number of items in Set 1:", sizeOfSet(set1))
```

```
J:\> python Level1Set.py
This Level 1 program makes a set
Set 1:{ 3, 5, 2, 6}
Number of items in Set 1: 4
J:\>
```

Your program should be called `Level1Set.py`. Leave it in the `J:` drive.

Level 2: Students finishing this level will be guaranteed to get **at least a C** for their continuous assessment grade.

For this level, you need to do everything from Level 1, plus:

- Write two functions, `union()` and `intersection()`. These two functions both take two sets as parameters and return a new set:
  - The **union** of two sets is a new set that consists of all the elements from the two old sets. Obviously the new set also has to obey set rules.
  - The **intersection** of two sets is a new set that consists only of the elements that appear in *both of the old sets*.

  Neither `union()` or `intersection()` should modify the old sets in any way.

The `main()` function for your program is shown below, and can be found in your `J:` drive as `Level2Main.py`. You are **NOT** allowed to change anything in `main()`.

```
def main():
    print("This Level 2 program deals with sets")
    set1 = makeSet("setData1.txt")
    set2 = makeSet("setData2.txt")
    print("Set 1:", end = "")
    printSet(set1)
    print("Set 2:", end = "")
    printSet(set2)

    print("Size of set1: ", sizeOfSet(set1))
    print("Size of set2: ", sizeOfSet(set2))
    unionSet = union(set1, set2)
    intersectionSet = intersection(set1, set2)
    print("Size of union: ", sizeOfSet(unionSet))
    print("Size of intersection: ", sizeOfSet(intersectionSet))
```

A sample run of your program is shown below:

```
J:\> python Level2Set.py
This Level 2 program deals with sets
Set 1:  28, 24, 98, 41, 72, 75, 43, 48, 34, 45, 97, 74, 29, 18, 30,
21, 37, 54, 36, 58, 76, 25, 14, 42, 44, 15, 16, 33, 90, 85, 3, 81,
19, 88, 92, 1, 31, 68
Set 2:  44, 62, 13, 14, 26, 12, 28, 93, 0, 98, 35, 40, 39, 8, 27, 79,
58, 42, 45, 23, 66, 20, 7, 21, 43, 47, 19, 82, 75, 70, 60, 69, 11, 2,
73, 38, 83, 63
Size of set1:  38
Size of set2:  38
Size of union:  65
Size of intersection:  11
J:\>
```

Your program should be called `Level2Set.py`. Leave it in the `J:` drive.

Level 3: Students finishing this level will be guaranteed to get **at least a B** for their continuous assessment grade.

For this level, you need to do everything required by Level 2, plus:

- Write a function, `sortSet()`. This function will take a set as a parameter and return a list, with the elements of the set sorted in ascending order (i.e. the smallest element in index 0, the second smallest element in index 1, etc). Your function should not modify the parameter set.

The `main()` function for your program is shown below, and can be found in your `J:` drive as `Level3Main.py`. You are **NOT** allowed to change anything in `main()`.

```
def main():
    print("This Level 3 program deals with sets")
    set1 = makeSet("setData1.txt")
    set2 = makeSet("setData2.txt")

    print("Size of set1: ", sizeOfSet(set1))
    print("Size of set2: ", sizeOfSet(set2))
    unionSet = union(set1, set2)
    intersectionSet = intersection(set1, set2)
    print("Size of union: ", sizeOfSet(unionSet))
    print("Size of intersection: ", sizeOfSet(intersectionSet))
    list1 = sortSet(intersectionSet)
    print("Sorted contents of intersection set:", list1)
```

A sample run of your program is shown below:

```
J:\> python Level3Set.py
This Level 3 program deals with sets
Size of set1:  38
Size of set2:  38
Size of union:  65
Size of intersection:  11
Sorted contents of intersection set:  [14, 19, 21, 28, 42, 43, 44,
45, 58, 75, 98]
J:\>
```

Your program should be called `Level3Set.py`. Leave it in the `J:` drive.

-End-