SSAD Assignment 1

Date: August 13, 2014

PART A

My Pacman:

Marks: 50

The first question is to write a python program that simulates a very basic version of Pacman Game.
Your program should have the following classes:

1. **Person**
2. **Pacman**
3. **Ghost**

Where Pacman and Ghost inherit from Person.
You have to implement it using OOP principles [**Inheritance**, **Polymorphism, Encapsulation** and **Modularity** are must]. The better code you write, more marks you get.

Don't worry! You don't have to apply any Artificial Intelligence here, you can control pacman's moves by pressing '**w**', '**a**', '**s**' and '**d**' keys for moving up, left, down and right respectively. Ghost's moves have to be **random**.

There have to be coins present in the space which are represented by '**C**', which on getting collected by Pacman increases your score.
Pacman is represented by '**P**' and Ghost by '**G**'.
Print the board after every input ('w', 's', 'a' and 'd').

You also need to have walls represented by '**X**' which can't be crossed by any person.

You are free to place pacman, ghost, coins and walls wherever you want initially.
Preferred board size is 35x15.
Least number of coins to be placed initially: 20
Least number of walls to be placed initially: 20

What should be the input?

1. Moves:
    a. '**w**' : up
    b. '**s**' : down
    c. '**a**' : left
    d. '**d**' : right
2. Quit '**q**'

What should be visible in the output?

1. Pacman **P**
2. Ghost **G**
3. Coins **C**
4. Walls **X**
5. Score
6. <Rest space represent using a dot> **.**

Some operations that you should have:

1. **checkGhost()**
   Checks if the current Pacman's position has a ghost on it and ENDS the game if true.
2. **collectCoin()**

3. **ghostPosition()**

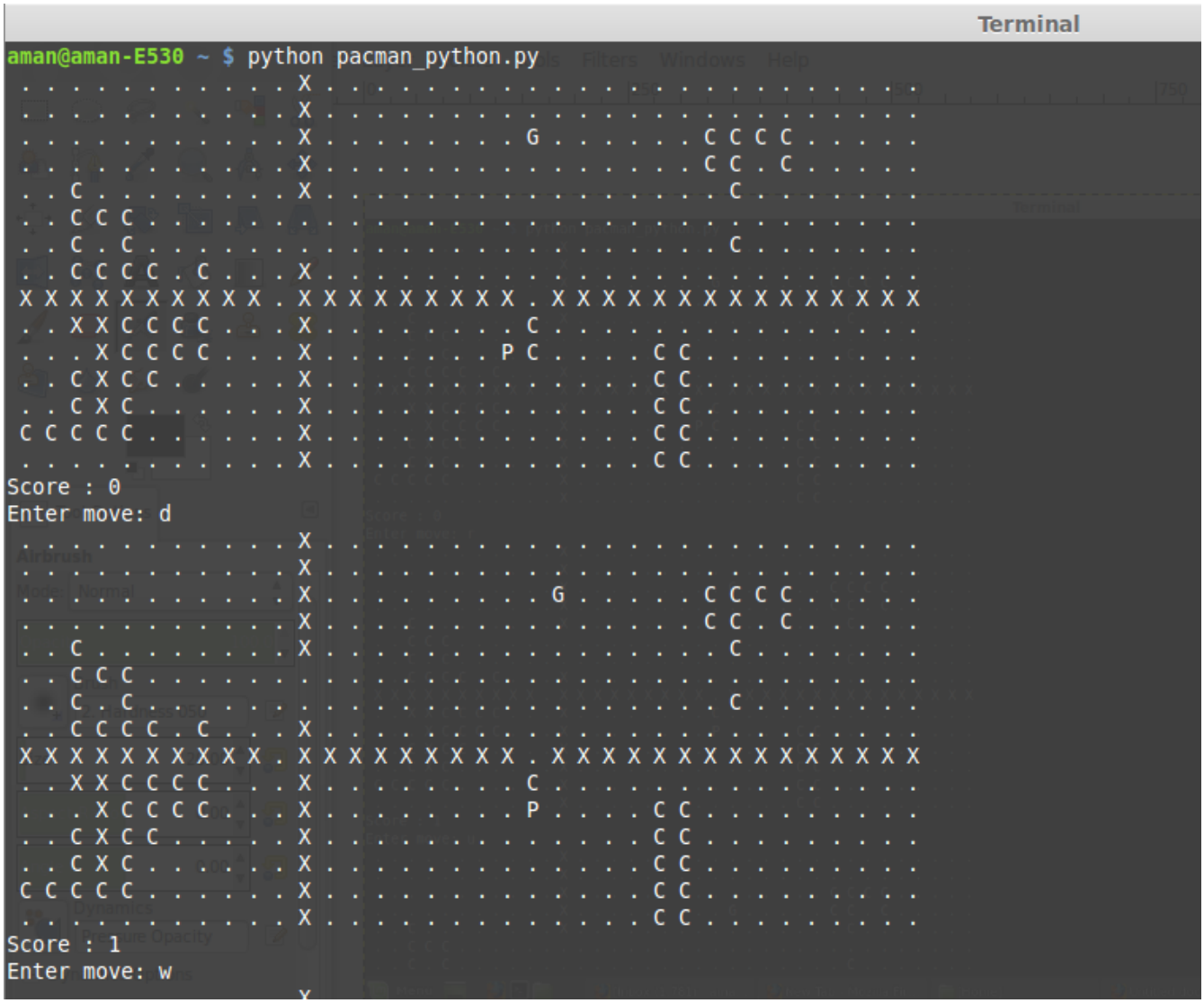
4. **checkWall()**

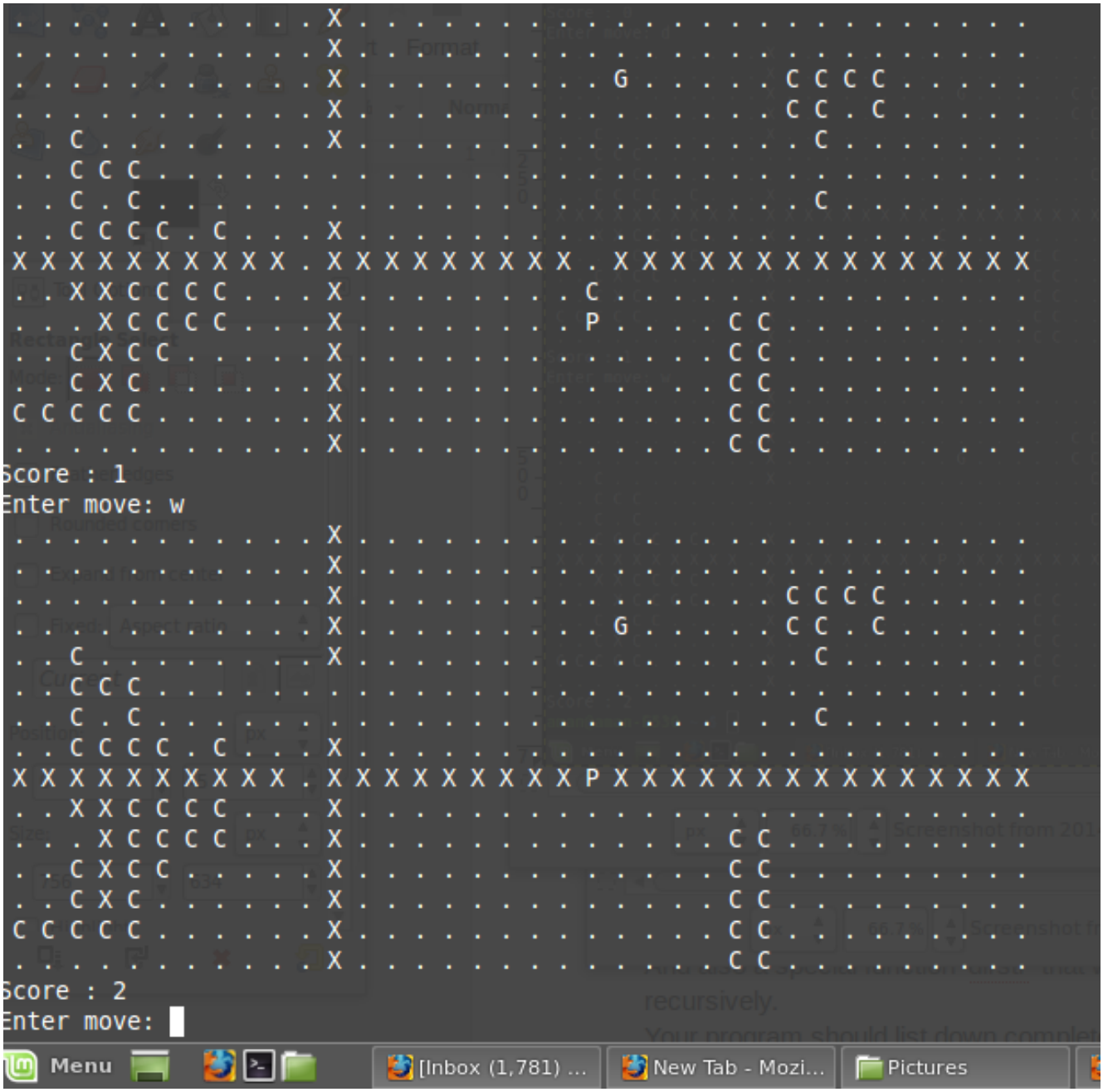
End the game when pacman gets eaten by ghost **or** user quits by pressing '**q**'.
Reload the board keeping the score intact when all the coins are taken.

*Bonus marks would be awarded for additional features(multiple ghosts,.. etc).
*Suggested OOP principles should be strictly followed.

Assessment Criteria :

OOP - Inheritance	**10 Marks**
OOP - Polymorphism	**10 Marks**
OOP - Modularity	**10 Marks**
OOP - Encapsulation	**10 Marks**
Functionalities of the Game	**10 Marks**
Bonus	**10 Marks**


Terminal
aman@aman-E530 ~ $ python pacman_python.py
Score : 0
Enter move: d
Score : 1
Enter move: w


Score : 1
Enter move: w
Score : 2
Enter move:
Menu
[Inbox (1,781) ...
New Tab - Mozi...
Pictures

PART B

My Commands:

Marks: 50

The second question is to write a python program that can simulate linux files and directory commands such as list(**ls**) ,copy(**cp**) ,move(**mv**) ,rename(**mv**) ,delete(**rm**),file details(**ls -l**),etc(Bonus marks for more functions of your choice).

And also a special function '**dirstr**' that will list directory structure of the present working directory recursively.
Your program should list down complete hierarchical structure of that directory.

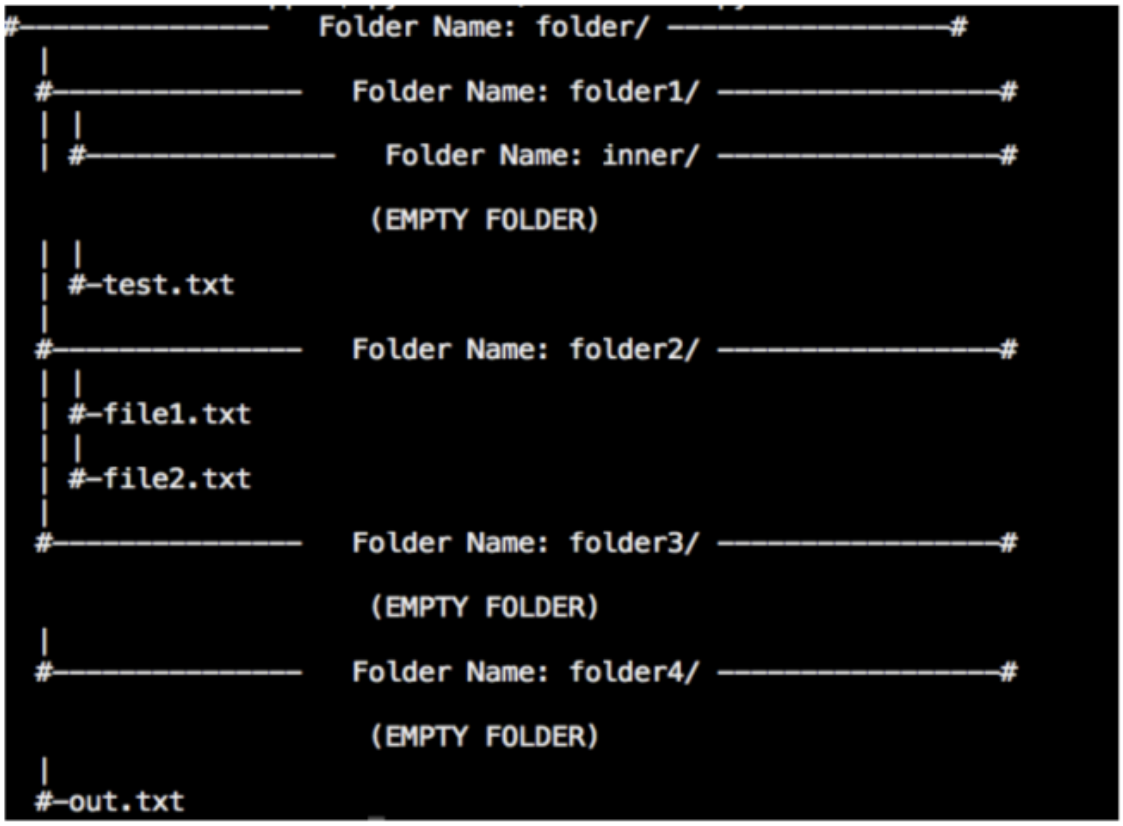
All these functions should work for both files and directories.
For example: 'rm -r Home' should delete the whole 'Home' directory recursively .

Format:
Your program should be run and accept all these commands in the same way bash does and output the same output as bash does.

For dirstr the desired output is given below output.

For Example:
If the directory structure is as follows:
(Current Directory : "folder")

Then your program should output this structure in the following manner for '**dirstr**':



Assessment Criteria :

Implementing ls, ls -l, mv, cp and rm.	**25 (5 each) Marks**
Implementing dirstr	**15 Marks**
Handling errors	**10 Marks**
Bonus	**10 Marks**

Note:

- In cases where a folder is empty it should be mentioned explicitly as (“EMPTY FOLDER”).
- Output format should be as close as possible to the given format
- Bonus points to implement more features to the above (like: file size, date modified etc).
- Error Handling should be done (for permission denied errors and others).
- Use of ‘os.system’ is **not** allowed

• **PLAGIARISM** IN ANY FORM WILL **NOT** BE TOLERATED . We will be running MOSS on all your assignments and with your seniors assignments as well, so please ensure nobody copies.
• Evaluation will be manual.

Upload Format:

•Name the files as “**A.py**” and “**B.py**” for part-A and part-B respectively.
•Put them in a folder named your Roll No “(eg :201401004)”
•Tar it and upload.