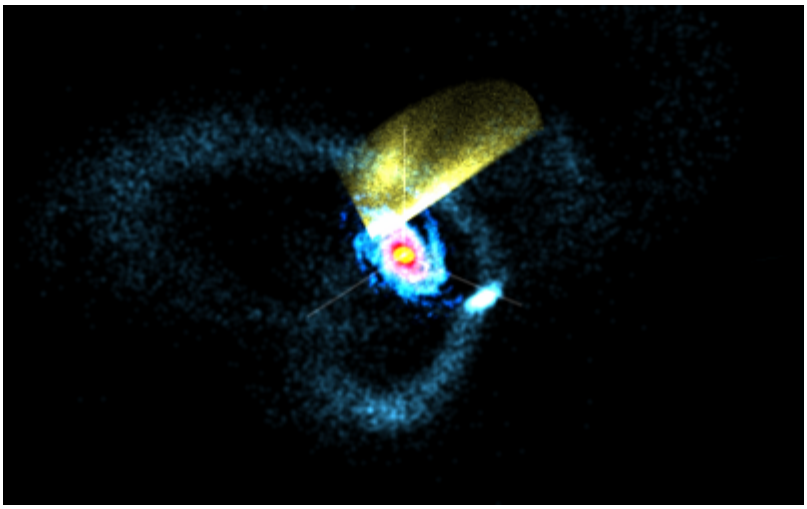
CSCI.4430/6430 Programming Languages Spring 2015
Programming Assignment #3

*This assignment is to be done either **individually** or **in pairs**. **Do not show your code to any other group** and **do not look at any other group's code**. Do not put your code in a public directory or otherwise make it public. However, you may get help from the TAs or the instructor. You are encouraged to use the LMS Discussions page to post problems so that other students can also answer/see the answers. **This assignment also must be done in an actor-based language. All assignments must provide a README** that explains how to run your code. If you use a language other than SALSA, you must explain how to install the necessary packages to run your code.*

You are to analyze three-dimensional data from the Sloan Digital Sky Survey, in particular, stars in our MilkyWay galaxy, for future human space colonization. Your program needs to compute the following:

- **Closest neighbours** Compute the pair of stars that minimize pairwise distance.
- **Farthest neighbours** Compute the pair of stars that maximize pairwise distance.
- **Ideal hub star** Compute the star which minimizes the maximal distance to any other star.
- **Ideal jail star** Compute the star which maximizes the minimal distance to any other star.
- **Ideal capital star** Compute the star which minimizes the average distance to all other stars.

For each computation, output multiple answers if there are ties. For example, when computing closest neighbours, if there are multiple pairs with the same minimum pairwise distance, output both pairs.



More information on Milky Way visualization can be found in the associated MilkyWay@Home forum.

You are given a stars text file with the first line giving the total number of stars in the file followed by one line per star representing each star's three dimensions <x,y,z> as X Y Z. Please remove duplicate entries in your program. Your output should look as follows:

```
d1  // minimal pairwise distance
s1 s2
...

d2  // maximal pairwise distance
s1 s2
s3 s2
...

d3 // minimum maximal distance
s3 s1
...

d4 // maximum minimal distance
s4
s5
...

d5 // minimal average distance
s1
...
```

where d_i denotes a distance and s_i denotes a star, which is represented as (x,y,z), corresponding to the x, y, and z coordinates of the star.

Part 1 - Concurrent Solution (60pt)

Write an actor-based solution to the space colonization problem.

Part 2 - Distributed Solution (40pt)

Write a distributed space colonization solution. Note that in this case, the actors must communicate over a network although you do not necessarily have to run each actor on a separate machine. If you are using SALSA, your solution will be distributed is you make use of universal actors.

Extra Credit - Mobile Actors Solution (10%)

Write an extension of your distributed space colonization solution so that actors can move to find better computational resources.

Other Possible Extensions (15%)

- Provide an analysis of sequential vs parallel execution performance.
- Provide an analysis of scalability.
- Develop a fault-tolerant distributed solution.

Time-Saving Hints

0. For reference, please see the SALSA webpage, including its FAQ. Read the tutorial and a comprehensive example illustrating distributed programming in SALSA.
1. `salsac` and `salsa` are UNIX aliases or Windows batch scripts that run `java` and `javac` with the expected arguments: See .cshrc for UNIX, and salsac.bat salsa.bat for Windows.
2. To run the distributed program, first, run the name server and the theaters:

```
[host0:dir0]$ wwcns [port number 0]
[host1:dir1]$ wwctheater [port number 1]
[host2:dir2]$ wwctheater [port number 2]
...
```

   where `wwcns` and `wwctheater` are UNIX aliases or Windows batch scripts: See .cshrc for UNIX, and wwcns.bat wwctheater.bat for Windows. Make sure that the theaters are run where the actor behavior code is available, that is, the `pa3` directory should be visible in directories: `host1:dir1` and `host2:dir2`. Then, run the distributed program as mentioned above.

3. The theaters all cache actor behaviors. Restart all the theaters each time changes are made to the code.
4. The module/behavior names in SALSA must match the directory/file hierarchical structure in the file system. e.g., a `Space` behavior should be in a relative path `pa3/Space.salsa`, and should start with the line `module pa3;`.
5. Messaging is asynchronous. `m1(...);m2(...);` does not imply `m1` occurs before `m2`.
6. Notice that in the code `m(...)@n(...);`, `n` is processed after m is executed, but not necessarily after messages sent *inside* `m` are executed. For example, if inside `m`, messages `m1` and `m2` are sent, in general, `n` could happen before `m1` and `m2`.
7. (Named) tokens **can only be used** as arguments to messages.

Due Date:

Received Time	Grade Modification
before Thursday, April 30th, 6:00PM	+10%
before Friday, May 1st, 6:00PM	no modification (on time)
before Saturday, May 2nd, 6:00PM	-10%
from Saturday, May 2nd, 6:00PM to Sunday, May 3rd, 6:00PM	-25%
after Sunday, May 3rd, 6:00PM	not accepted

Grading: The assignment will be graded mostly on correctness, but code clarity / readability will also be a factor (comment, comment, comment!).

Submission Requirements: Please submit a ZIP file with your code, including a README file. In the README file, place the names of each group member. Your README file should also have a list of specific features / bugs in your solution. Your ZIP file should be named with your LMS user name(s) as the filename, either userid1.zip or userid1_userid2.zip. Only submit one assignment per

pair via LMS.