Assignment 2 | Distributed Systems

`Bhuvanesh Sridharan`
`2018113002`

Q1

The spawner spawns P-1 processes with them being in `life()` function which acts as the self referencing infinite loop for the program. The Spawner also appends itself to the list of processes so that the message it had sent, returns to itself.

In `life()`, The process waits for a data of type `{Sender, TokenValue, Procs, ProcsList, OutFile}`. And this subprocess then send the TokenValue to the next process in the `Procs` List of variable.

Q2

Bellman ford algorithm has been used to distribute the workload and do this work parallely.

Why the solution is truly parallel:

- Instead of a list to store {Vertex, Distance} values a **map** has been used so that retrieving and update operations are `O(1)` instead of `O(N)` like in the case of Lists.
- The subprocesses spawned have the `subprocess_life()` in which they simply wait for the input of type `{Server, Edgelist, CurMap}` and then relaxe the edges they have got in the `Edgelist` variable.
- They return **only those node value pairs** which have changed since the last time and not the whole {vertex, value} map.
- Then each process gets another set of edges to relax for the next iteration.
- Hence work of each subprocess is truly atomised in the sense that the same subprocess might receive different set of edges in the next iteration and the code would still function properly which allows for an increase in the number of processes dynamically during the runtime.

Time Complexity:

- Worst case complexity is still `O(V*E)`.
- With the distributed implementation, we will get some constant factor improvisation.

- Best case complexity is `O(V*E/P)` when there won't be much changes in each relaxation iteration.
