Question 2

One control the simulator gives you is the fork percentage ,controlled by the -f flag. The higher it is, the more likely the next action is a fork; the lower it is, the more likely the action is an exit. Run the simulator with a large number of actions (e.g., -a 100) and vary the fork percentage from 0.1 to 0.9. What do you think the resulting final process trees will look like as the percentage changes? Check your answer with -c.

The higher the percentage the bigger the tree.

```
                           Process Tree:
                               a

Action: a forks b
                               a
                               └── b
Action: b forks c
                               a
                               └── b
                                   └── c
Action: a forks d
                               a
                               ├── b
                               │   └── c
                               └── d
Action: d forks e
                               a
                               ├── b
                               │   └── c
                               └── d
                                   └── e
Action: b forks f
                               a
                               ├── b
                               │   ├── c
                               │   └── f
                               └── d
                                   └── e
```

Question 4

One interesting thing to note is what happens when a child exits; what happens to its children in the process tree? To study this, let’s create a specific example: ./fork.py -A a+b,b+c,c+d,c+e,c-. This example has process ’a’ create ’b’, which in turn creates ’c’, which then creates ’d’ and ’e’. However, then, ’c’ exits. What do you think the process tree should like after the exit? What if you use the -R flag? Learn more about what happens to orphaned pro- cesses on your own to add more context.

All children of c become children of a since a is the root. With the -R the children become the children of the youngest living parent.

```
                           Process Tree:
                               a

Action: a forks b
                               a
                               └── b
Action: b forks c
                               a
                               └── b
                                   └── c
Action: c forks d
                               a
                               └── b
                                   └── c
                                       └── d
Action: c forks e
                               a
                               └── b
                                   └── c
                                       ├── d
                                       └── e
Action: c EXITS
                               a
                               ├── b
                               ├── d
                               └── e
```