Chapter 5 HW

1. Run./fork.py -s 10 and see which actions are taken. Can you predict what the process tree looks like at each step? Use the -c flag to check your answers. Try some different random seeds (-s) or add more actions (-a) to get the hang of it.
A:
First, a has two children first (b and c), and then c exits and now a has only one child(b). Then a has two new children(d and e).

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.
A:
Higher the percentage is given, more layer the tree has. If the percentage is 0.9 or 0.8, the processes generated by the fork() won't exist, and there will be more and more layer in the tree. On the other hand, if the percentage is lower, like 0.1, every child process can be easily ended, and the tree will be likely a root with few children.

3. Now, switch the output by using the -t flag (e.g., run ./fork.py -t). Given a set of process trees, can you tell which actions were taken?
A:
Yes.

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 processes on your own to add more context.
A:
If a process exists, then its all children will become the root's new children. Therefore, for the given example, d and e will become a's children.

5. One last flag to explore is the -F flag, which skips intermediate steps and only asks to fill in the final process tree. Run ./fork.py -F and see if you can write down the final tree by looking at the series of actions generated. Use different random seeds to try this a few times.
A:
Yes.

6. Finally, use both -t and -F together. This shows the final process tree, but then asks you to fill in the actions that took place. By looking at the tree, can you determine the exact actions that took place? In which cases can you tell? In which can't you tell? Try some different random seeds to delve into this question.
A:
If the processes number except a is as same as the number of action, then it means all actions are fork() and we can determine the exact actions. However, if the number actions is larger than the number of processes, it means there are exit actions and we may not predict the exact actions.