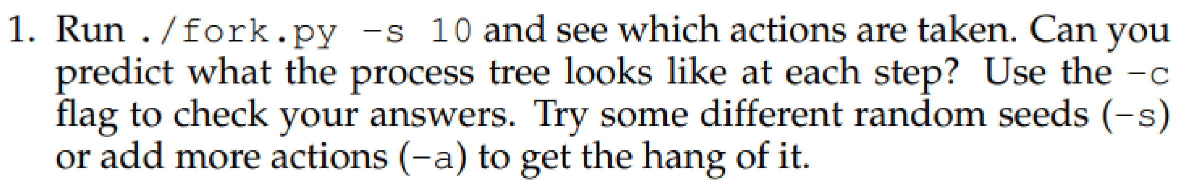
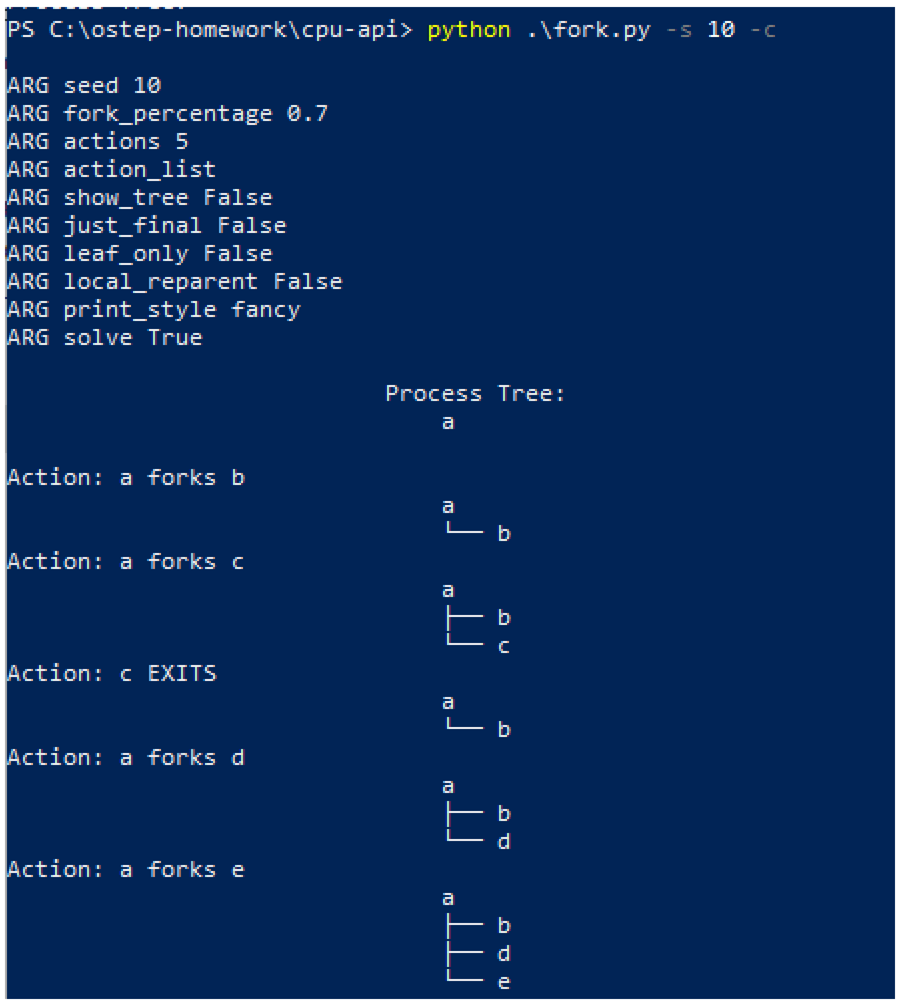
【OS】Day9(2)

【Ch2】Process API Homework

Question 1

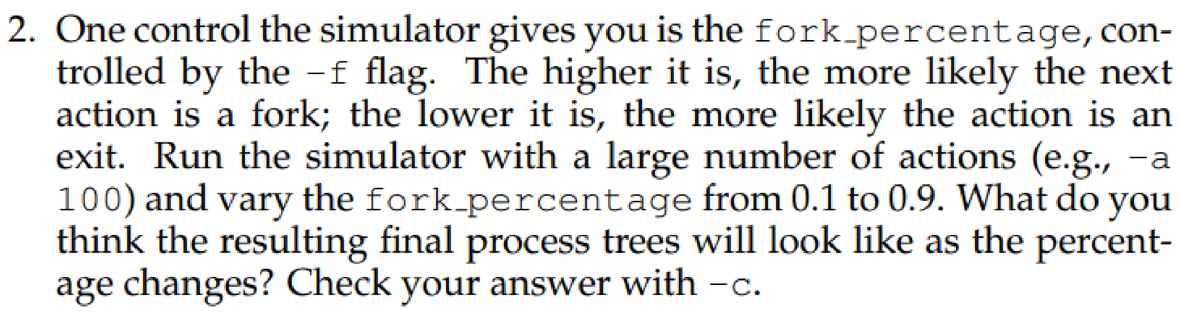


1. Run `./fork.py -s 10` and see which actions are taken. Can you predict what the process tree looks like at each step? Use the `-c` flag to check your answers. Try some different random seeds (`-s`) or add more actions (`-a`) to get the hang of it.



```
PS C:\ostep-homework\cpu-api> python .\fork.py -s 10 -c

ARG seed 10
ARG fork_percentage 0.7
ARG actions 5
ARG action_list
ARG show_tree False
ARG just_final False
ARG leaf_only False
ARG local_reparent False
ARG print_style fancy
ARG solve True

                           Process Tree:
                               a

Action: a forks b
                               a
                               └── b

Action: a forks c
                               a
                               ├── b
                               └── c

Action: c EXITS
                               a
                               └── b

Action: a forks d
                               a
                               ├── b
                               └── d

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

Question 2

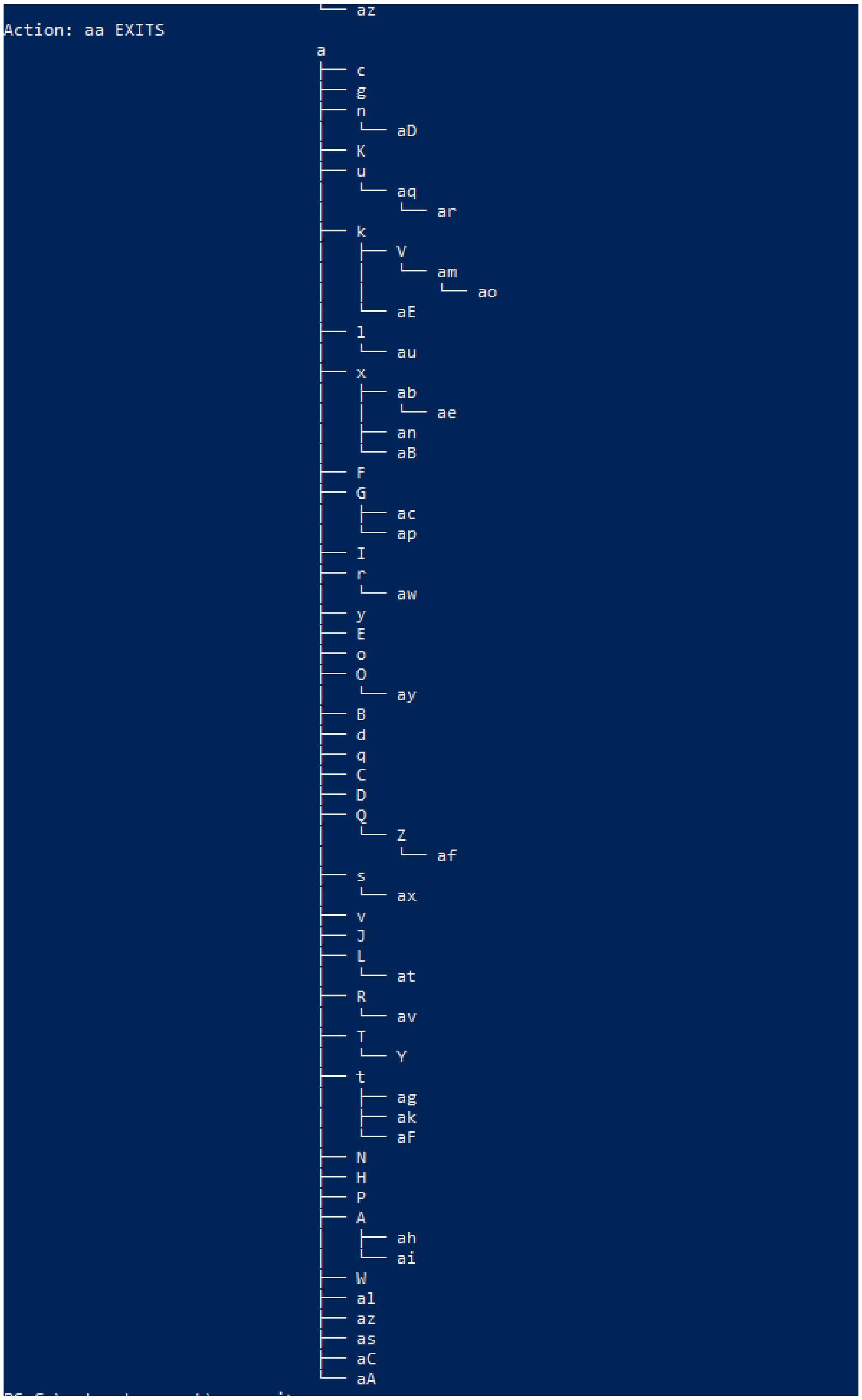


2. One control the simulator gives you is the `fork_percentage`, controlled by the `-f` flag. The higher it is, the more likely the next action is a fork; the lower it is, the more likely the action is an exit. Run the simulator with a large number of actions (e.g., `-a 100`) and vary the `fork_percentage` 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`.

If we provide a large `fork_percentage`, the next action is less likely to be `exit`. Thus, as the number of processes get larger, the process tree also grows really large

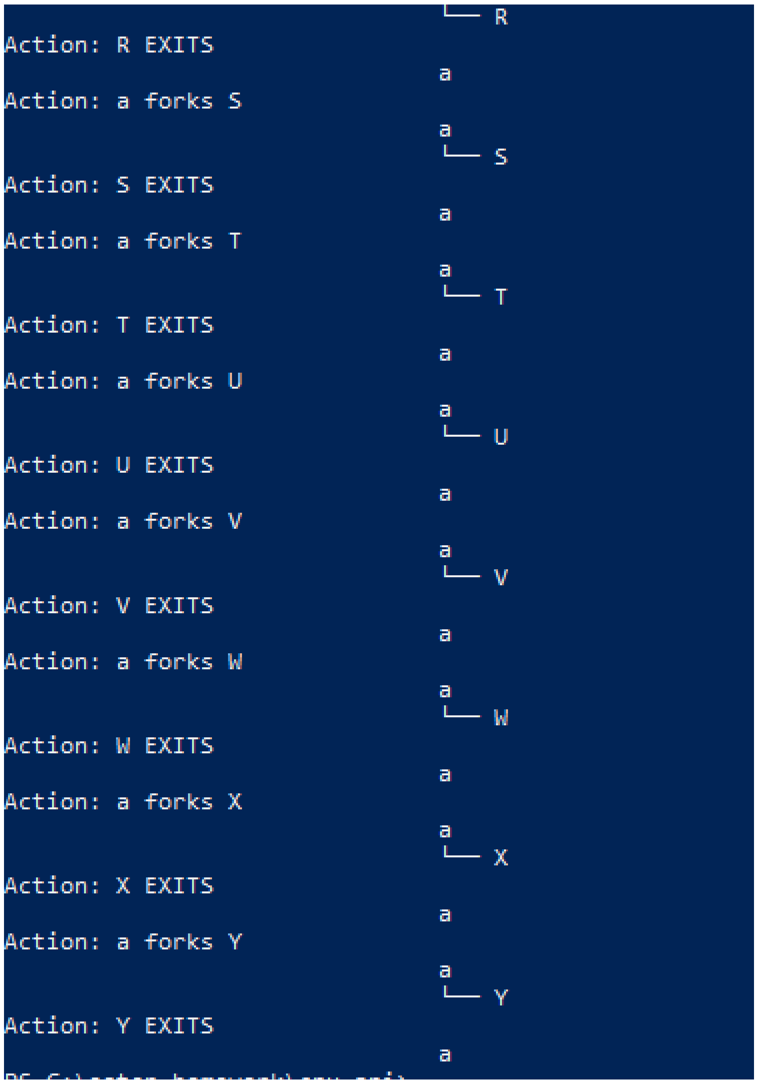
If we provide a small number, then the process will be terminated more frequently and thus result into a smaller process tree.

Result of a large `fork_percentage` number:



```
                               └── az
Action: aa EXITS
                               a
                               ├── c
                               ├── g
                               ├── n
                               │   └── aD
                               ├── K
                               ├── u
                               │   └── aq
                               │       └── ar
                               ├── k
                               │   ├── V
                               │   │   └── am
                               │   │       └── ao
                               │   └── aE
                               ├── l
                               │   └── au
                               ├── x
                               │   ├── ab
                               │   │   └── ae
                               │   ├── an
                               │   └── aB
                               ├── F
                               ├── G
                               │   ├── ac
                               │   └── ap
                               ├── I
                               ├── r
                               │   └── aw
                               ├── y
                               ├── E
                               ├── o
                               ├── O
                               │   └── ay
                               ├── B
                               ├── d
                               ├── q
                               ├── C
                               ├── D
                               ├── Q
                               │   └── Z
                               │       └── af
                               ├── s
                               │   └── ax
                               ├── v
                               ├── J
                               ├── L
                               │   └── at
                               ├── R
                               │   └── av
                               ├── T
                               │   └── Y
                               ├── t
                               │   ├── ag
                               │   ├── ak
                               │   └── aF
                               ├── N
                               ├── H
                               ├── P
                               ├── A
                               │   ├── ah
                               │   └── ai
                               ├── W
                               ├── al
                               ├── az
                               ├── as
                               ├── aC
                               └── aA
```

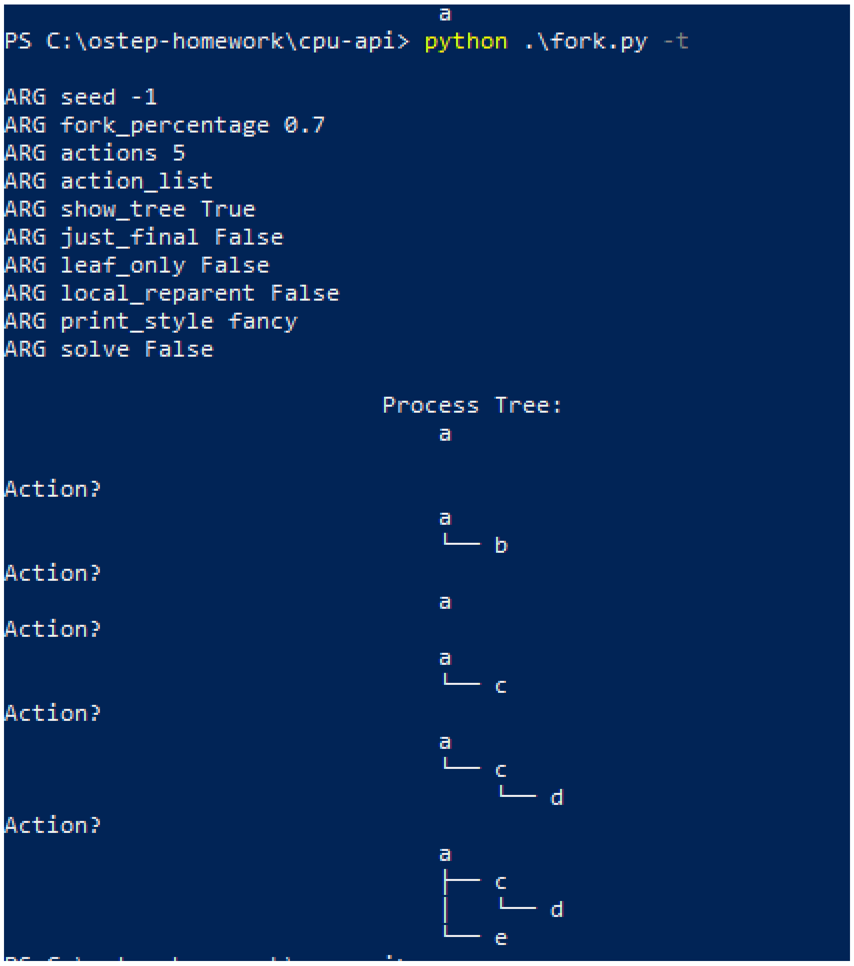
Result of a small `fork_percentage` number:



```
                               └── R
Action: R EXITS
                               a
Action: a forks S
                               a
                               └── S
Action: S EXITS
                               a
Action: a forks T
                               a
                               └── T
Action: T EXITS
                               a
Action: a forks U
                               a
                               └── U
Action: U EXITS
                               a
Action: a forks V
                               a
                               └── V
Action: V EXITS
                               a
Action: a forks W
                               a
                               └── W
Action: W EXITS
                               a
Action: a forks X
                               a
                               └── X
Action: X EXITS
                               a
Action: a forks Y
                               a
                               └── Y
Action: Y EXITS
                               a
```

Question 3

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?

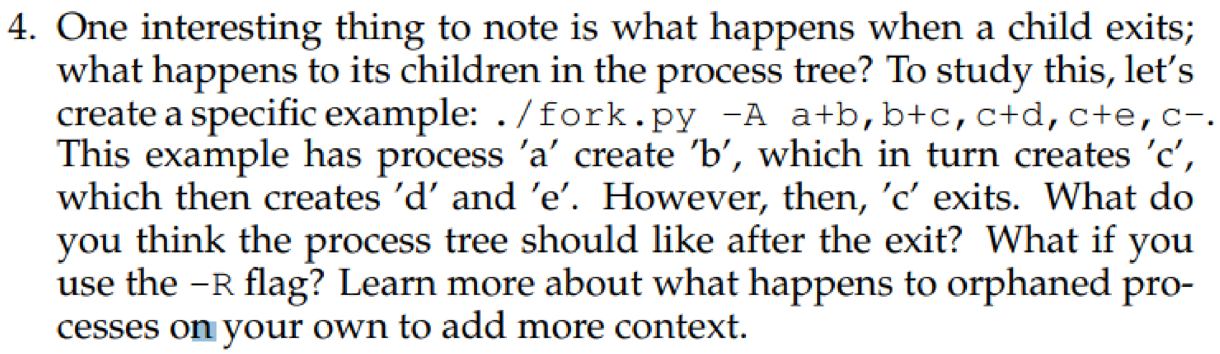


```
                              a
PS C:\ostep-homework\cpu-api> python .\fork.py -t

ARG seed -1
ARG fork_percentage 0.7
ARG actions 5
ARG action_list
ARG show_tree True
ARG just_final False
ARG leaf_only False
ARG local_reparent False
ARG print_style fancy
ARG solve False

                           Process Tree:
                               a

Action?
                               a
                               └── b
Action?
                               a

Action?
                               a
                               └── c

Action?
                               a
                               └── c
                                   └── d
Action?
                               a
                               ├── c
                               │   └── d
                               └── e
```

Initial process: a

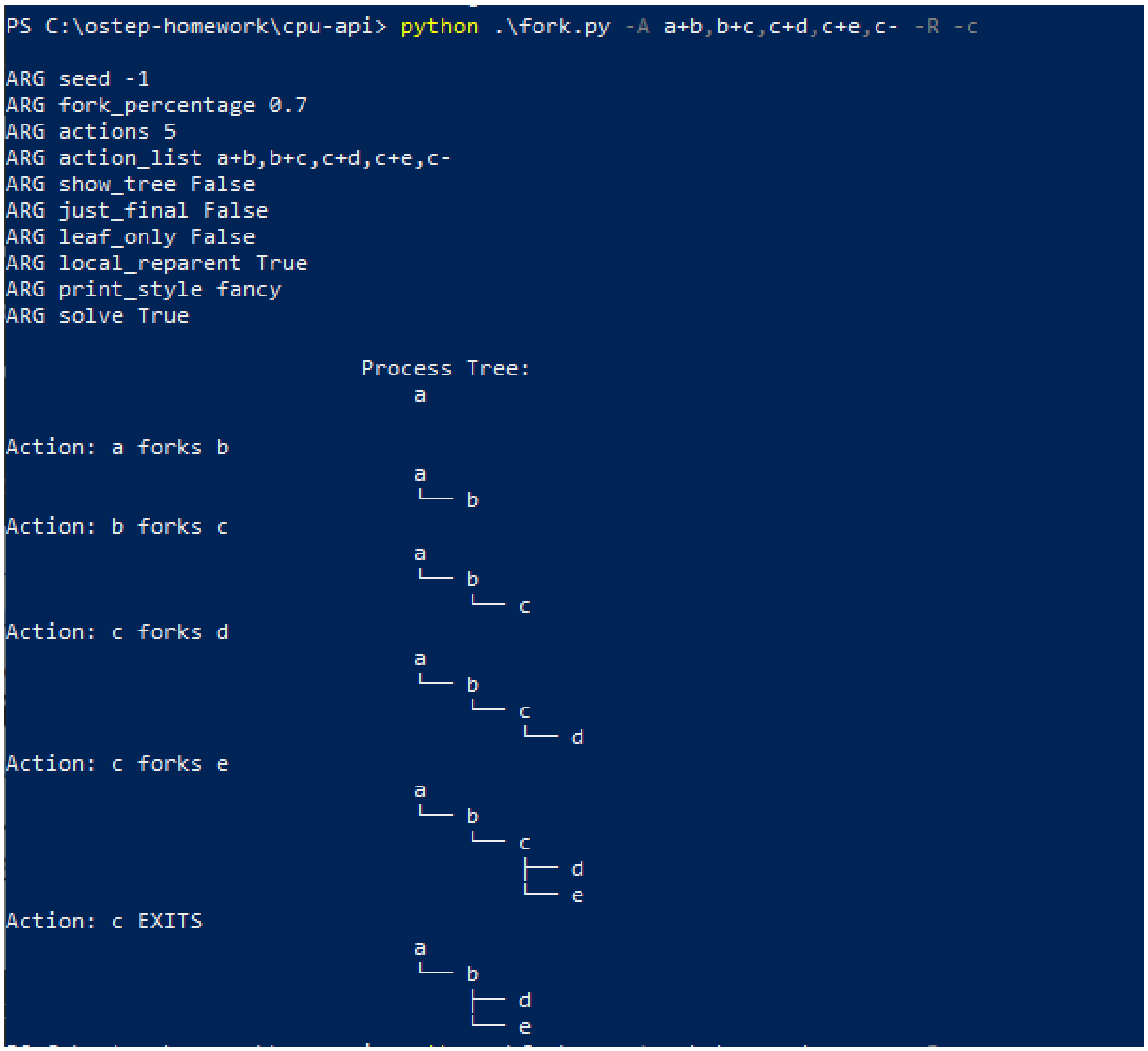
1. a fork b
2. b exits
3. a fork c
4. c fork d
5. a fork e

Question 4



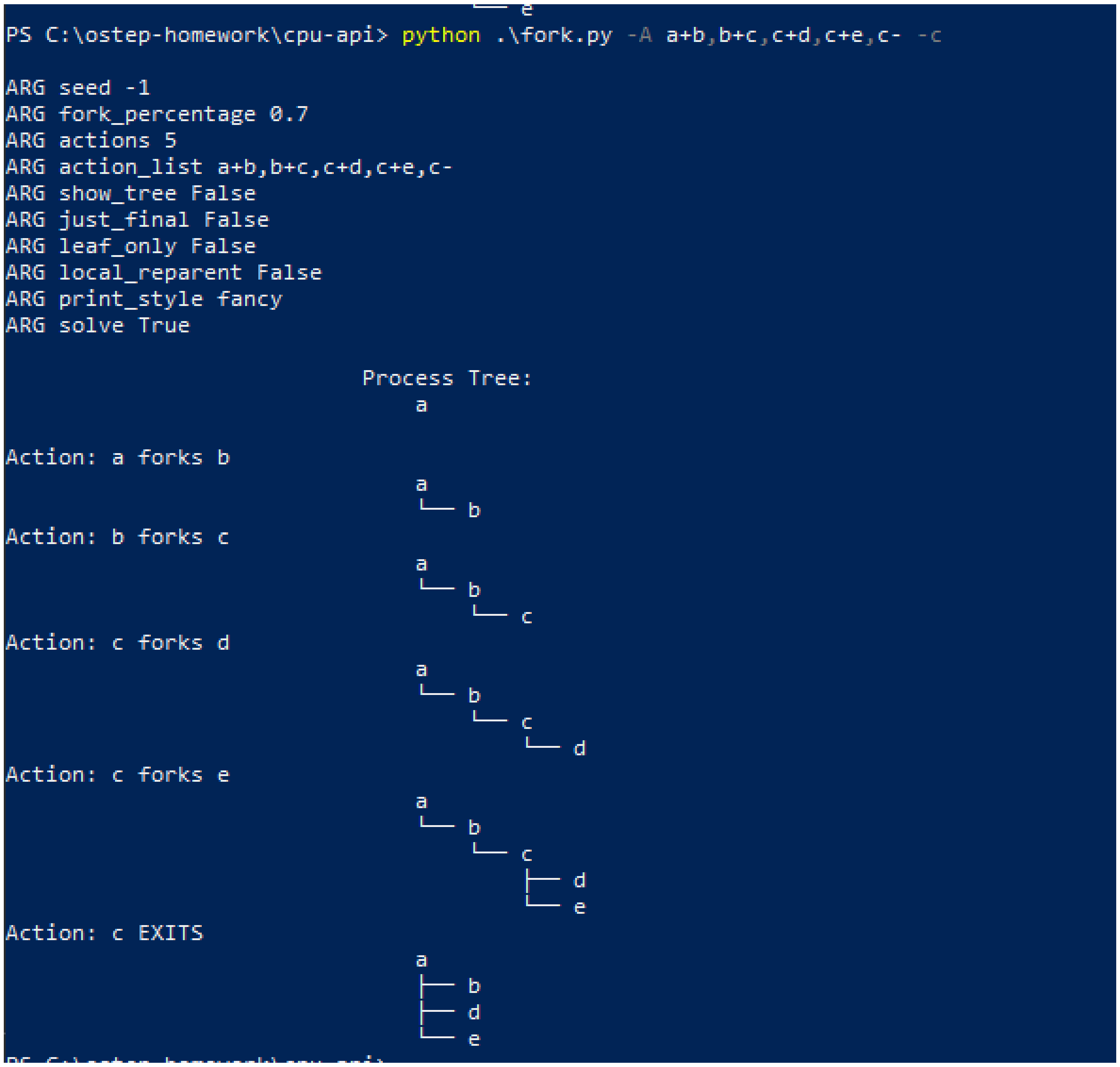
4. One interesting thing to note is what happens when a child exits; what happens to its children in the process tree? To study this, let's create a specific example: `./fork.py -A a+b,b+c,c+d,c+e,c-`. This example has process 'a' create 'b', which in turn creates 'c', which then creates 'd' and 'e'. 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.

The -R options turns on the `local_replacement` option. When turned on, the children processes will become the children process of the nearest parent process.



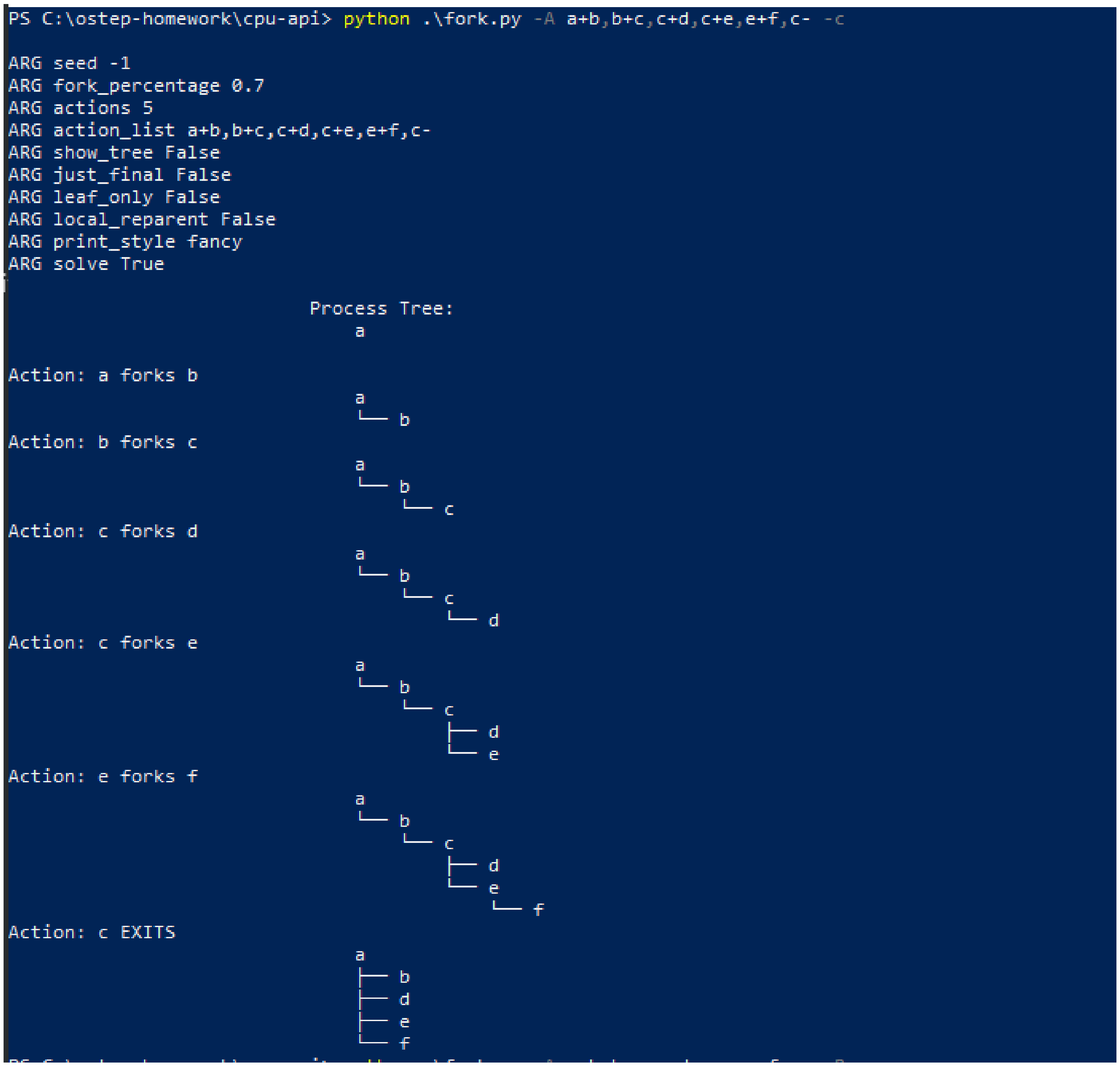
```
PS C:\ostep-homework\cpu-api> python .\fork.py -A a+b,b+c,c+d,c+e,c- -R -c

ARG seed -1
ARG fork_percentage 0.7
ARG actions 5
ARG action_list a+b,b+c,c+d,c+e,c-
ARG show_tree False
ARG just_final False
ARG leaf_only False
ARG local_reparent True
ARG print_style fancy
ARG solve True

                           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
```

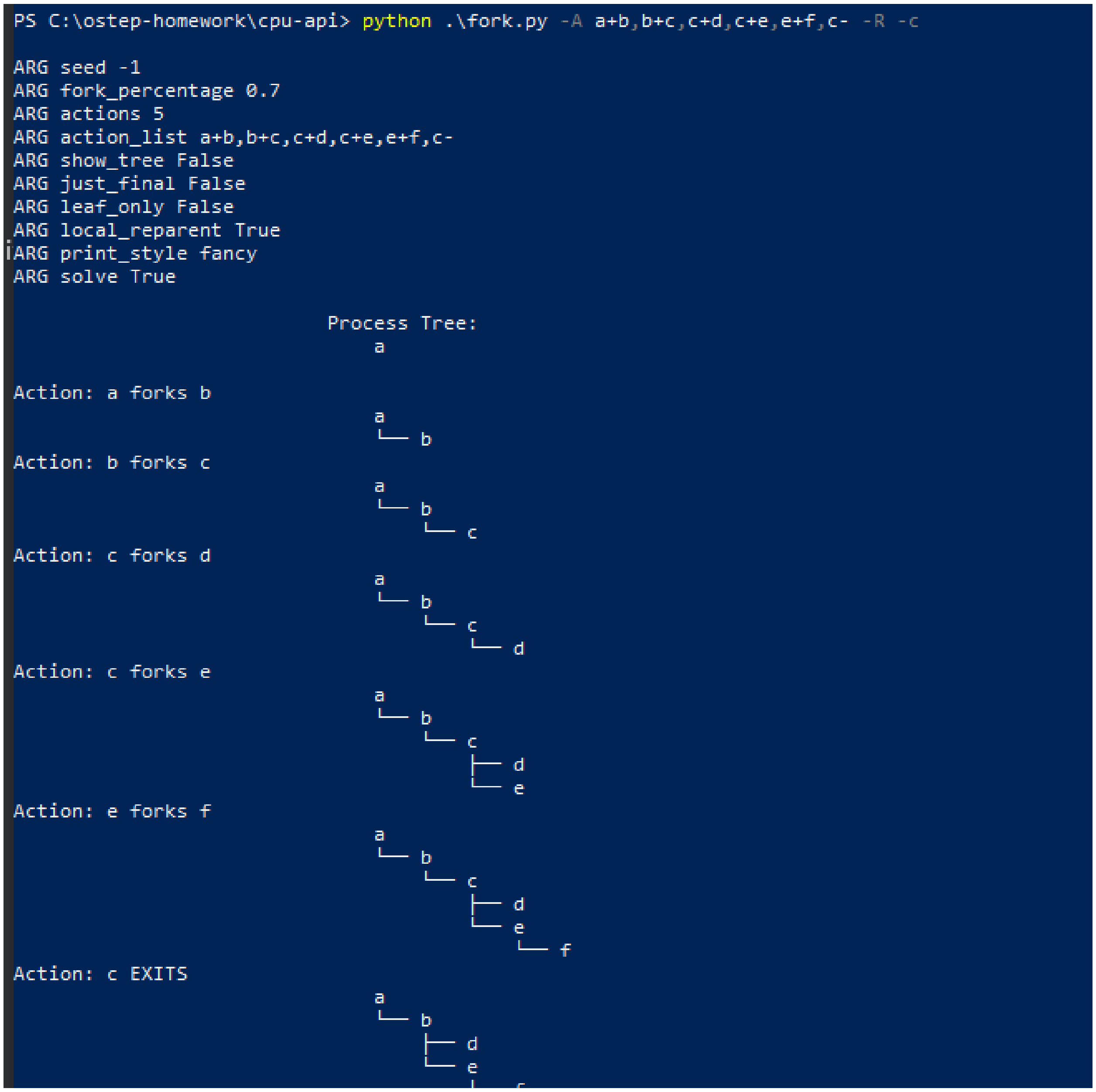
However, if we do not have it turned on, the children processes will be appended to the initial process.



```
                                                       └── e
PS C:\ostep-homework\cpu-api> python .\fork.py -A a+b,b+c,c+d,c+e,c- -c

ARG seed -1
ARG fork_percentage 0.7
ARG actions 5
ARG action_list a+b,b+c,c+d,c+e,c-
ARG show_tree False
ARG just_final False
ARG leaf_only False
ARG local_reparent False
ARG print_style fancy
ARG solve True

                           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
```

Note: If the children process also creates children process, then those children processes will also become children processes of the initial process if `local_replacement` *is not on.*


```
PS C:\ostep-homework\cpu-api> python .\fork.py -A a+b,b+c,c+d,c+e,e+f,c- -c

ARG seed -1
ARG fork_percentage 0.7
ARG actions 5
ARG action_list a+b,b+c,c+d,c+e,e+f,c-
ARG show_tree False
ARG just_final False
ARG leaf_only False
ARG local_reparent False
ARG print_style fancy
ARG solve True

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



```
PS C:\ostep-homework\cpu-api> python .\fork.py -A a+b,b+c,c+d,c+e,e+f,c- -R -c

ARG seed -1
ARG fork_percentage 0.7
ARG actions 5
ARG action_list a+b,b+c,c+d,c+e,e+f,c-
ARG show_tree False
ARG just_final False
ARG leaf_only False
ARG local_reparent True
ARG print_style fancy
ARG solve True

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

Question 6

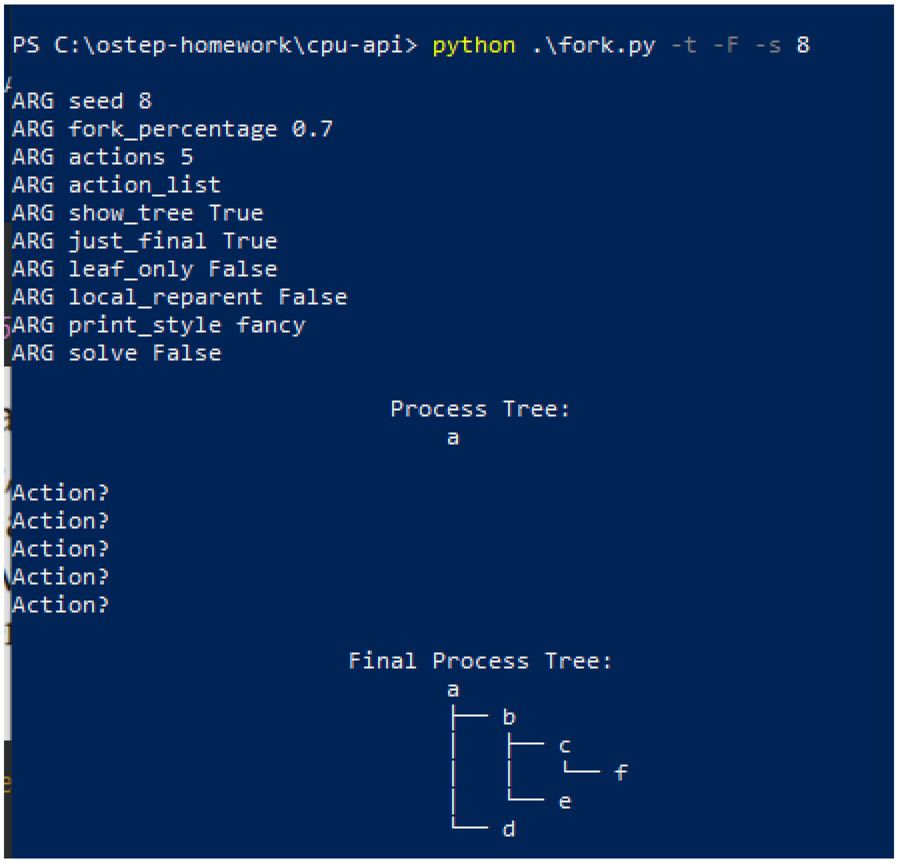
6. Finally, use both `-t` and `-F` together. This shows the final process tree, but then asks you to fill in the actions that took place. By looking at the tree, can you determine the exact actions that took place? In which cases can you tell? In which can't you tell? Try some different random seeds to delve into this question.

If all of the letters are provided in the alphabetic order, then we can conclude the order in which they are created.

If some letter is missing, we know that exit is called. Thus it would diverge into the following two occasions:

- If `local_reaplacement` is on, then we know where the children processes were.
- If `local_replacement` isn't on, we cannot conclude where the children processes were.

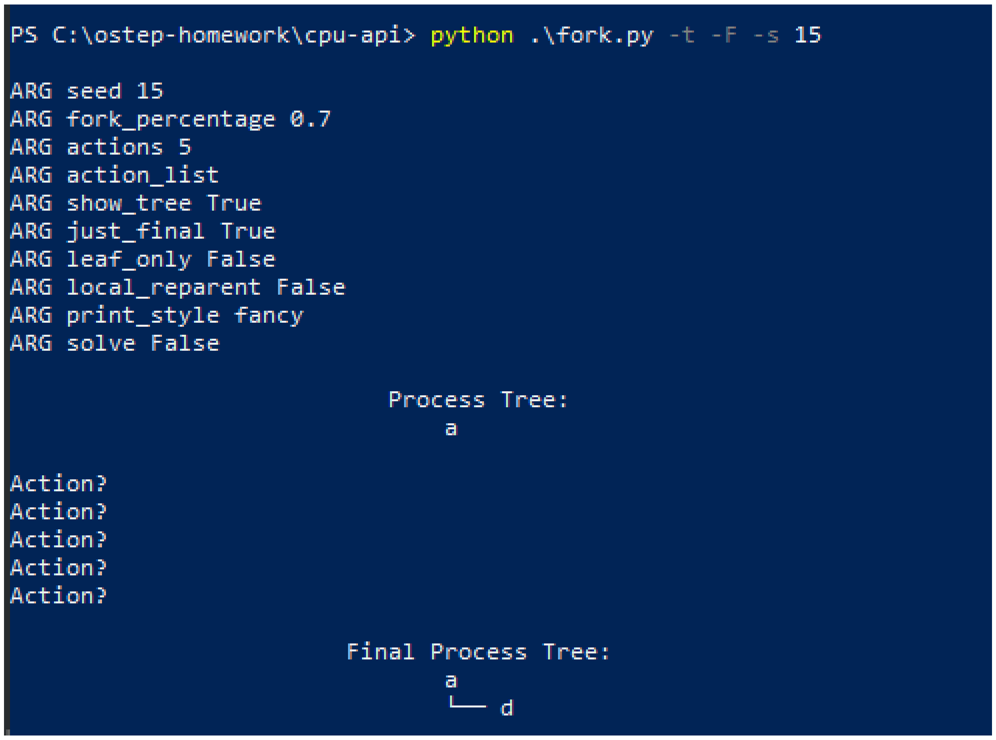
The result of this is obvious:



```
PS C:\ostep-homework\cpu-api> python .\fork.py -t -F -s 8

ARG seed 8
ARG fork_percentage 0.7
ARG actions 5
ARG action_list
ARG show_tree True
ARG just_final True
ARG leaf_only False
ARG local_reparent False
ARG print_style fancy
ARG solve False

                           Process Tree:
                               a

Action?
Action?
Action?
Action?
Action?

                        Final Process Tree:
                               a
                               ├── b
                               │   ├── c
                               │   │   └── f
                               │   └── e
                               └── d
```

1. a fork b
2. b fork c
3. a fork d
4. b fork e
5. c fork f

However, not to this:



```
PS C:\ostep-homework\cpu-api> python .\fork.py -t -F -s 15

ARG seed 15
ARG fork_percentage 0.7
ARG actions 5
ARG action_list
ARG show_tree True
ARG just_final True
ARG leaf_only False
ARG local_reparent False
ARG print_style fancy
ARG solve False

                           Process Tree:
                               a

Action?
Action?
Action?
Action?
Action?

                        Final Process Tree:
                               a
                               └── d
```

Several possible answer:

1. a fork b
2. a fork c
3. a fork d
4. b exit
5. c exit

Or:

1. a fork b
2. b fork c
3. c fork d
4. b exit(c and d now both become children of a)
5. c exit