



# Operating Systems

## Processes-Part3

Seyyed Ahmad Javadi

[sajavadi@aut.ac.ir](mailto:sajavadi@aut.ac.ir)

Fall 2024

# Fork Example



output



```
i = 0
import os

i=0
while i < 2:
    print(i)
    os.fork()
    i += 1
```

# Fork Example



output



```
i = 0
import os

i=0
while i < 2:
    print(i)
    os.fork()
    i += 1
```

# Fork Example

0

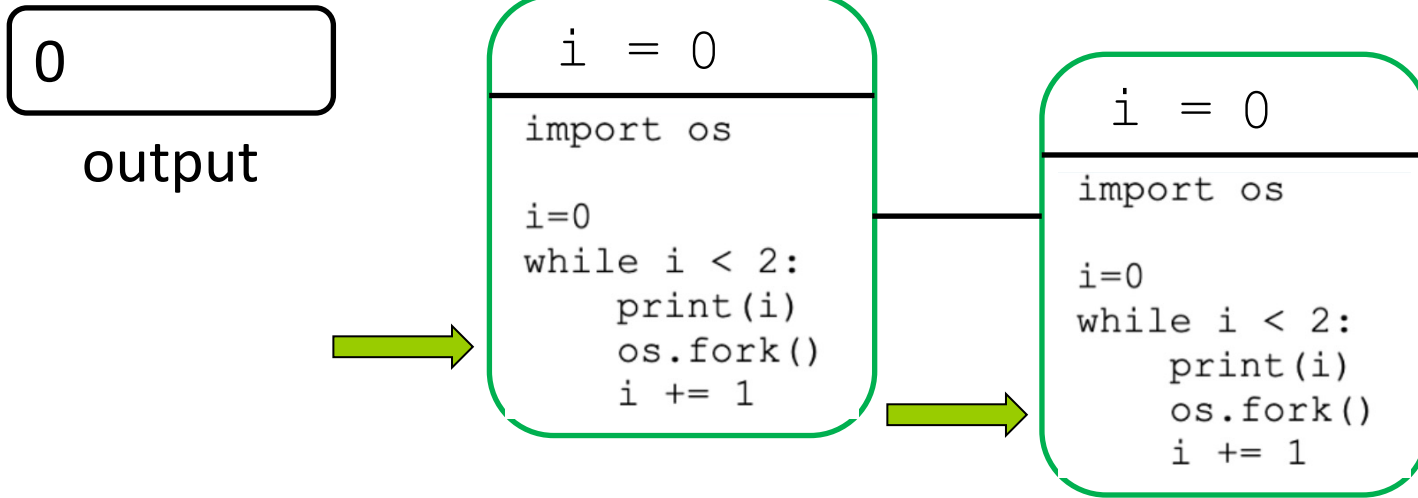
output



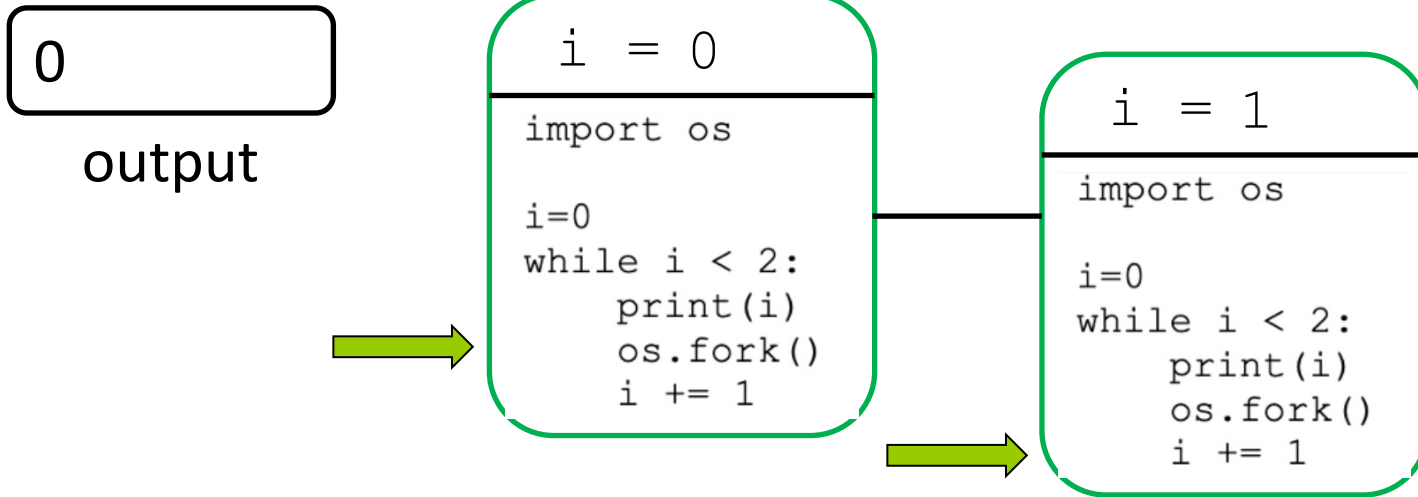
```
i = 0
import os

i=0
while i < 2:
    print(i)
    os.fork()
    i += 1
```

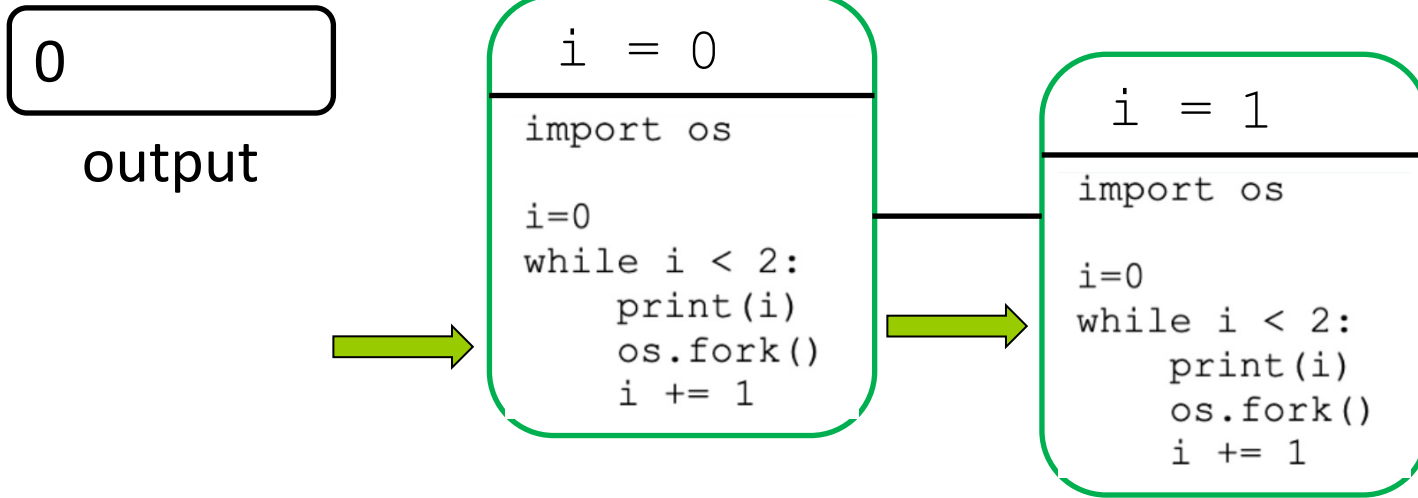
# Fork Example



# Fork Example



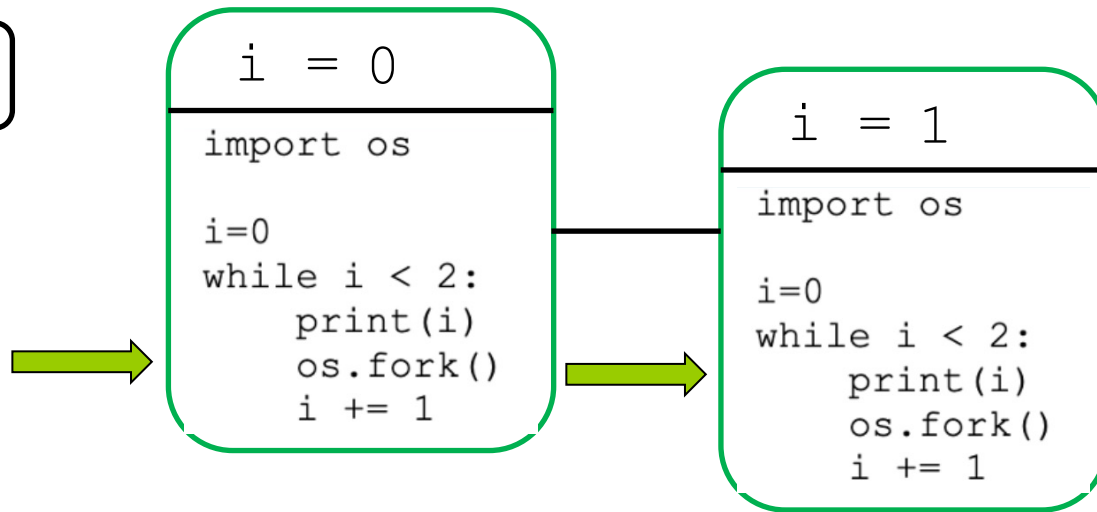
# Fork Example



# Fork Example

01

output

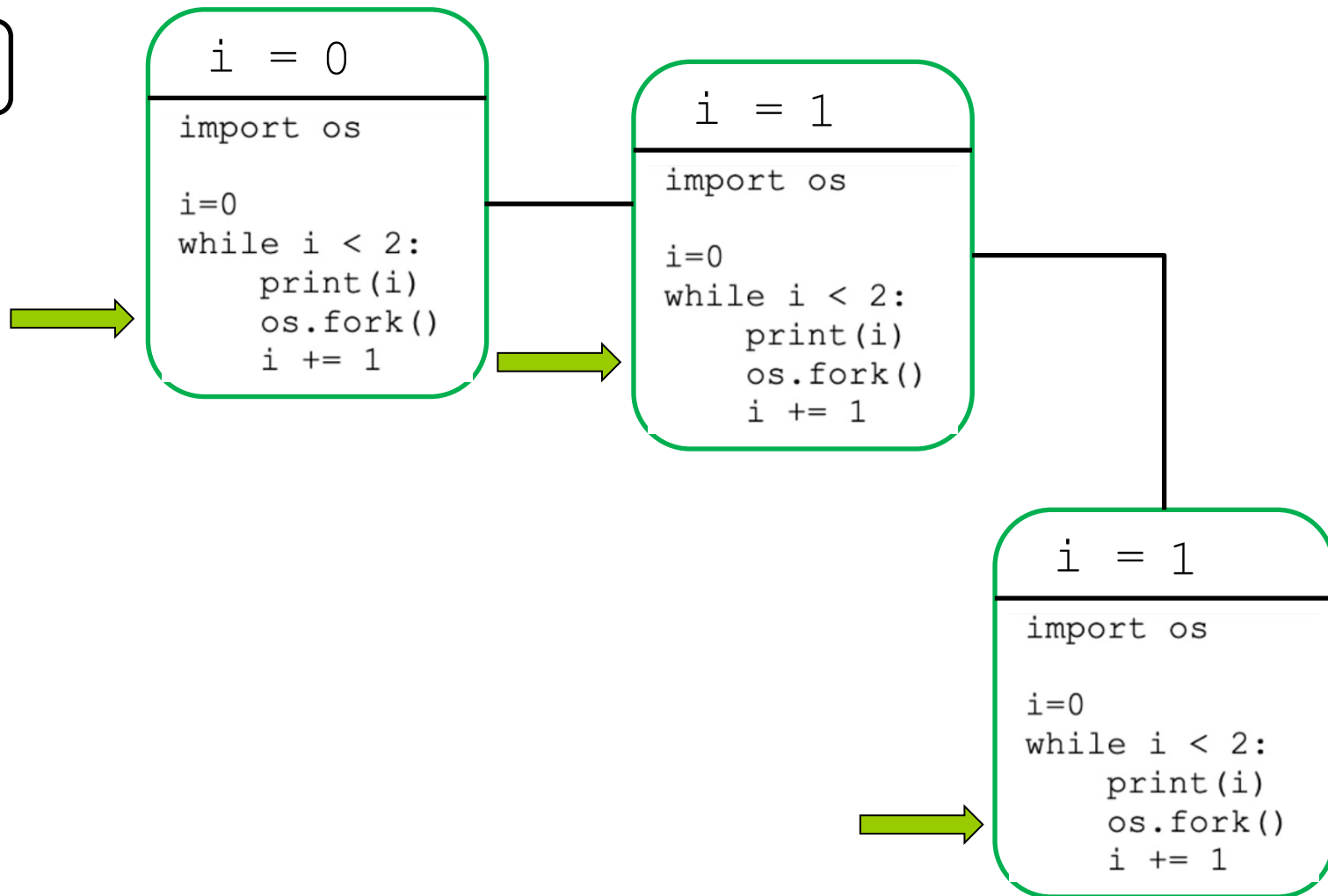




# Fork Example

01

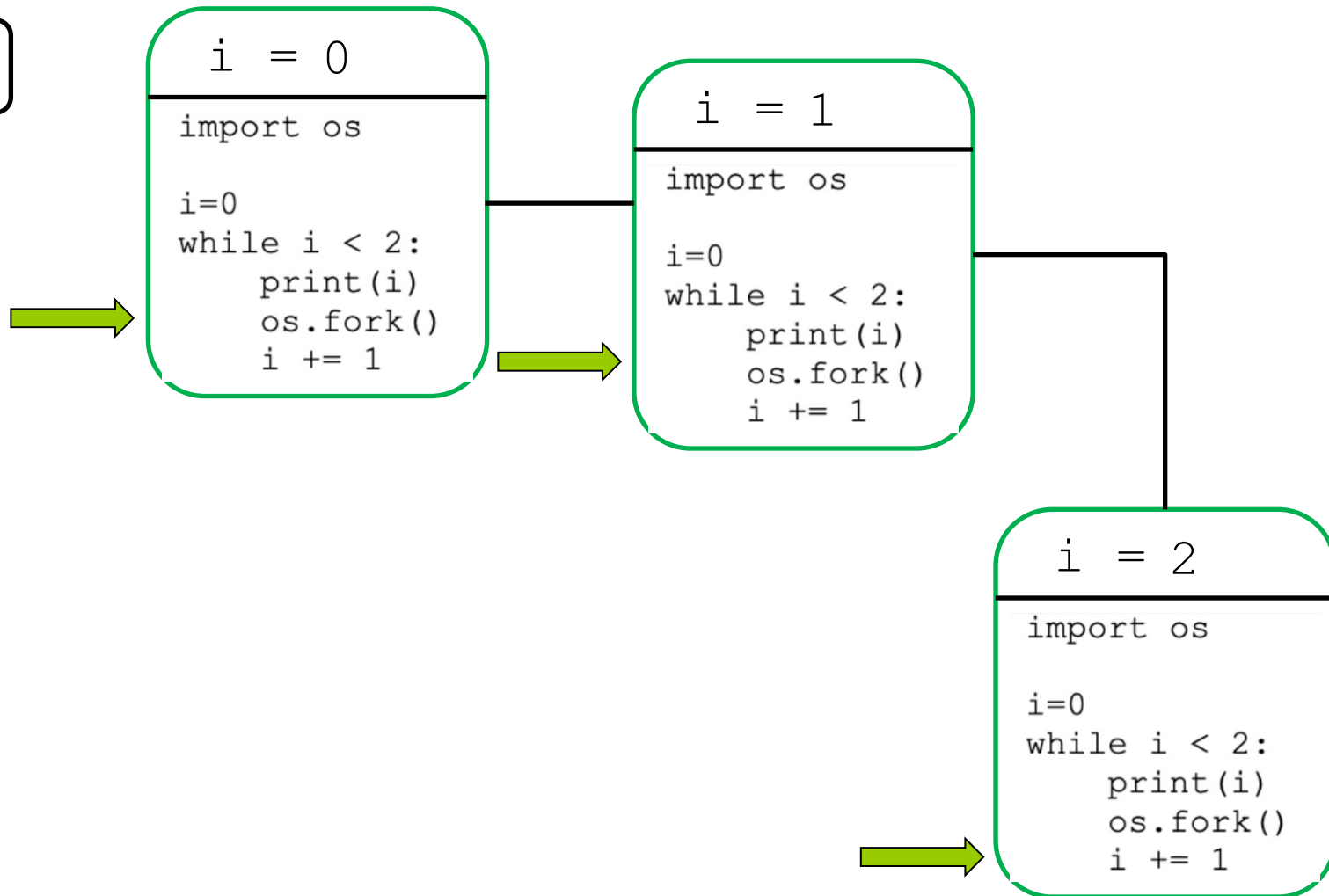
output



# Fork Example

01

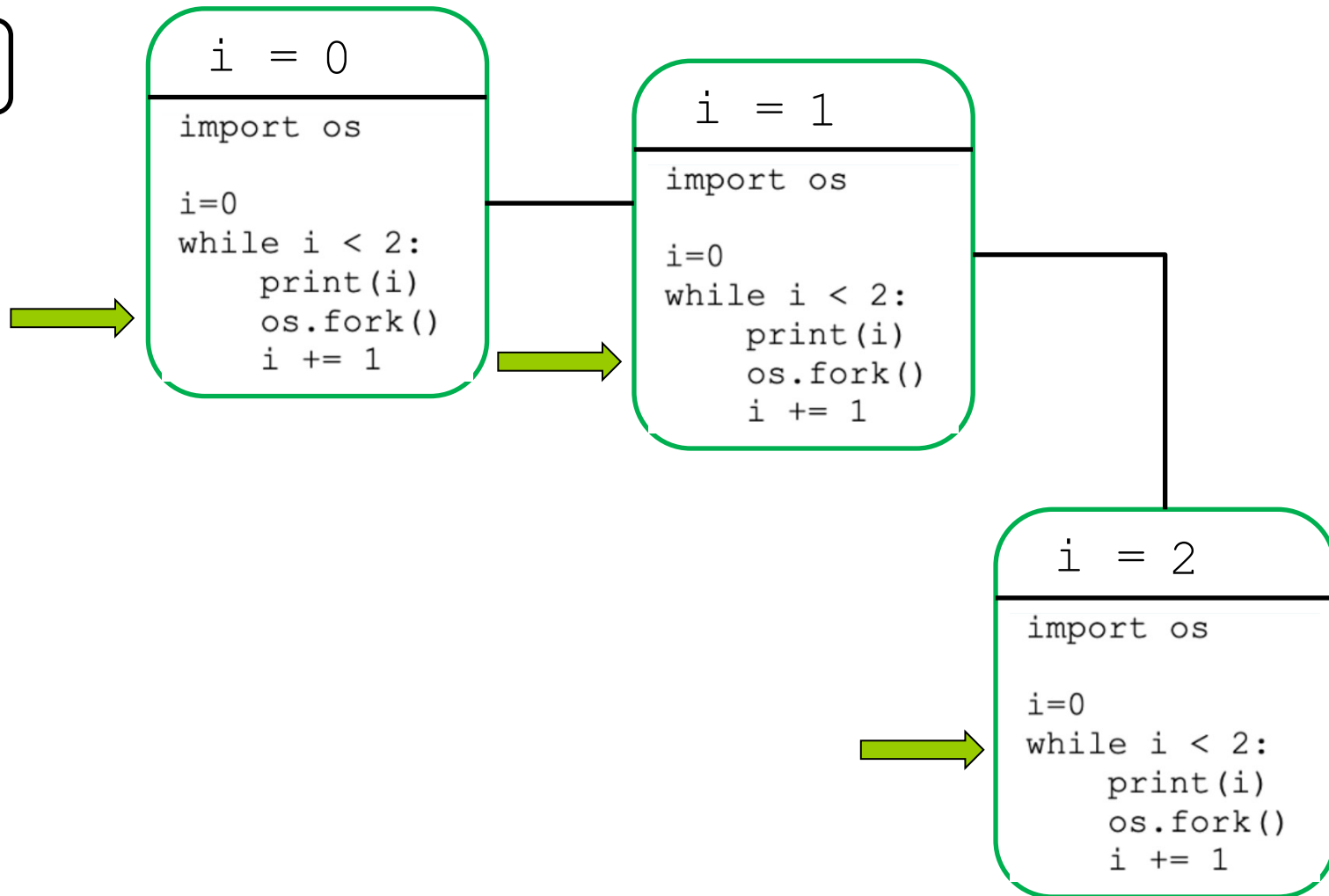
output



# Fork Example

01

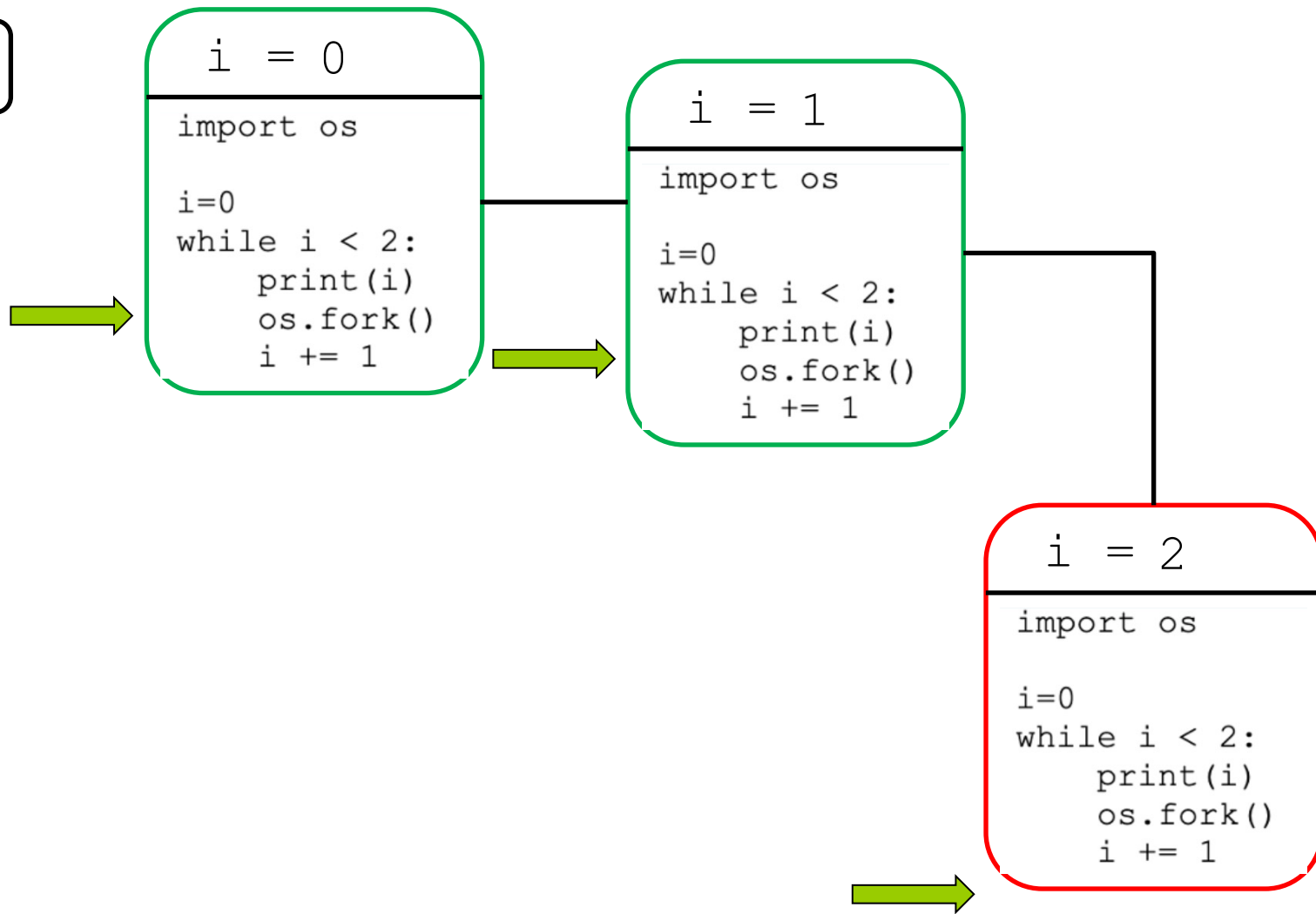
output



# Fork Example

01

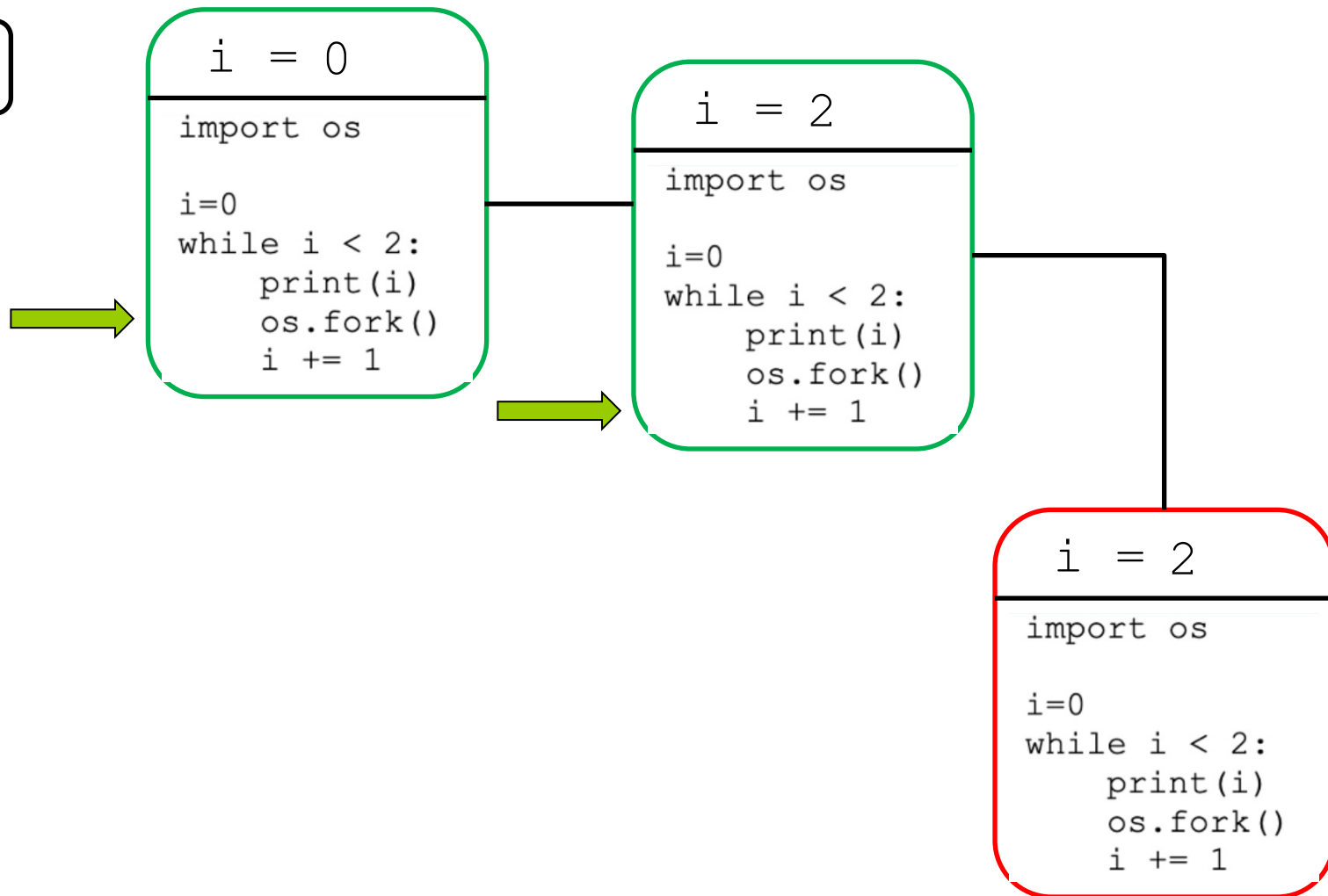
output



# Fork Example

01

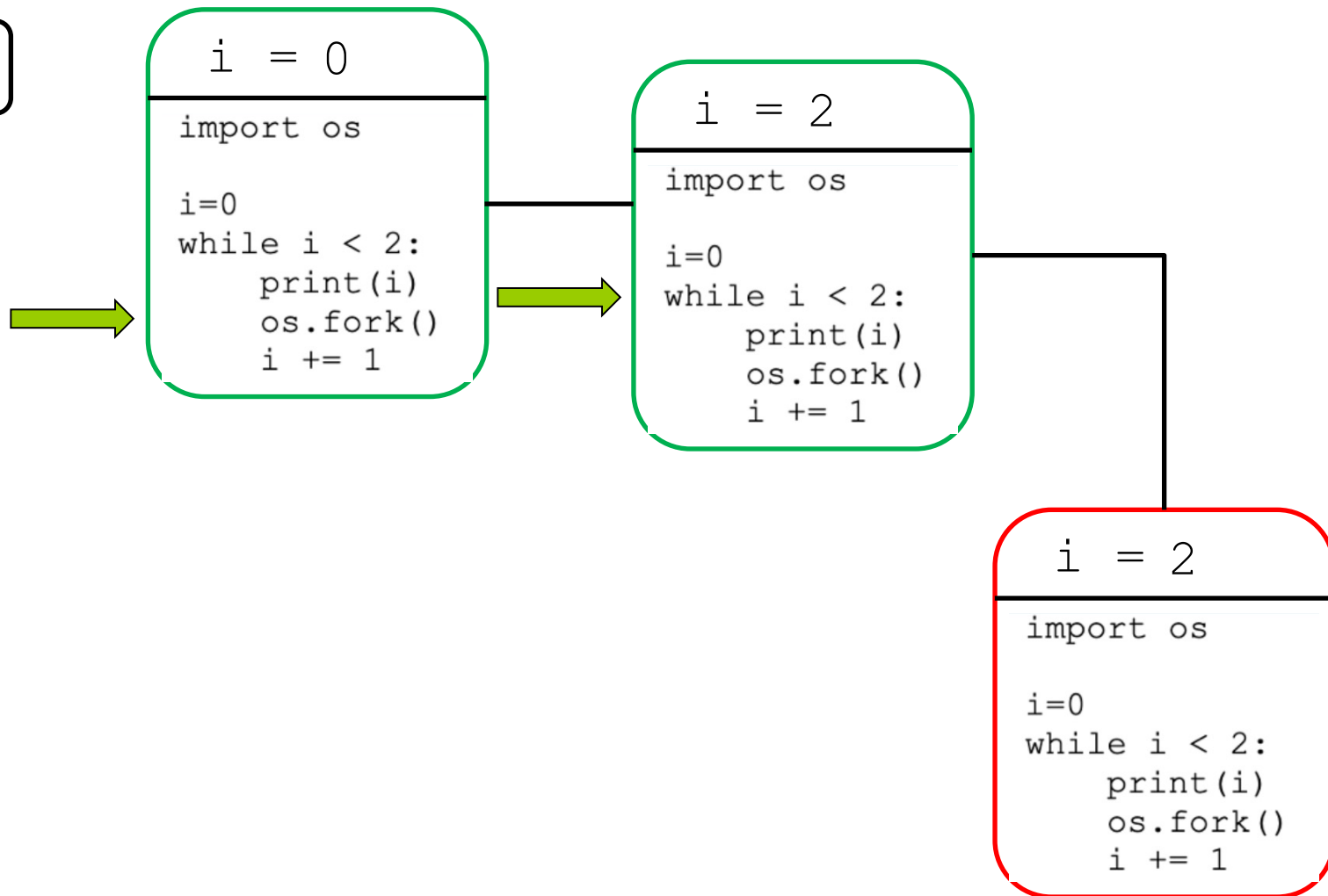
output



# Fork Example

01

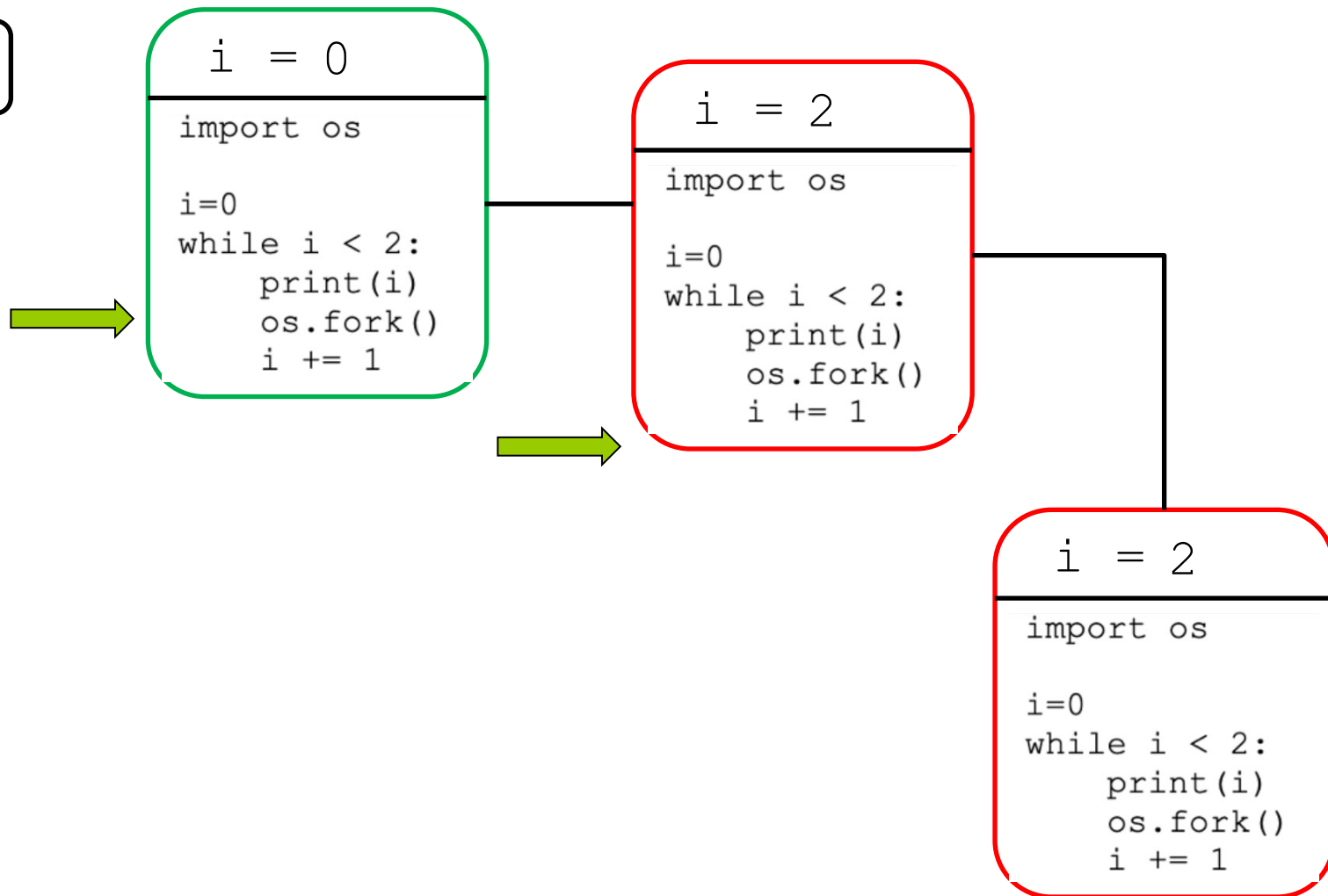
output



# Fork Example

01

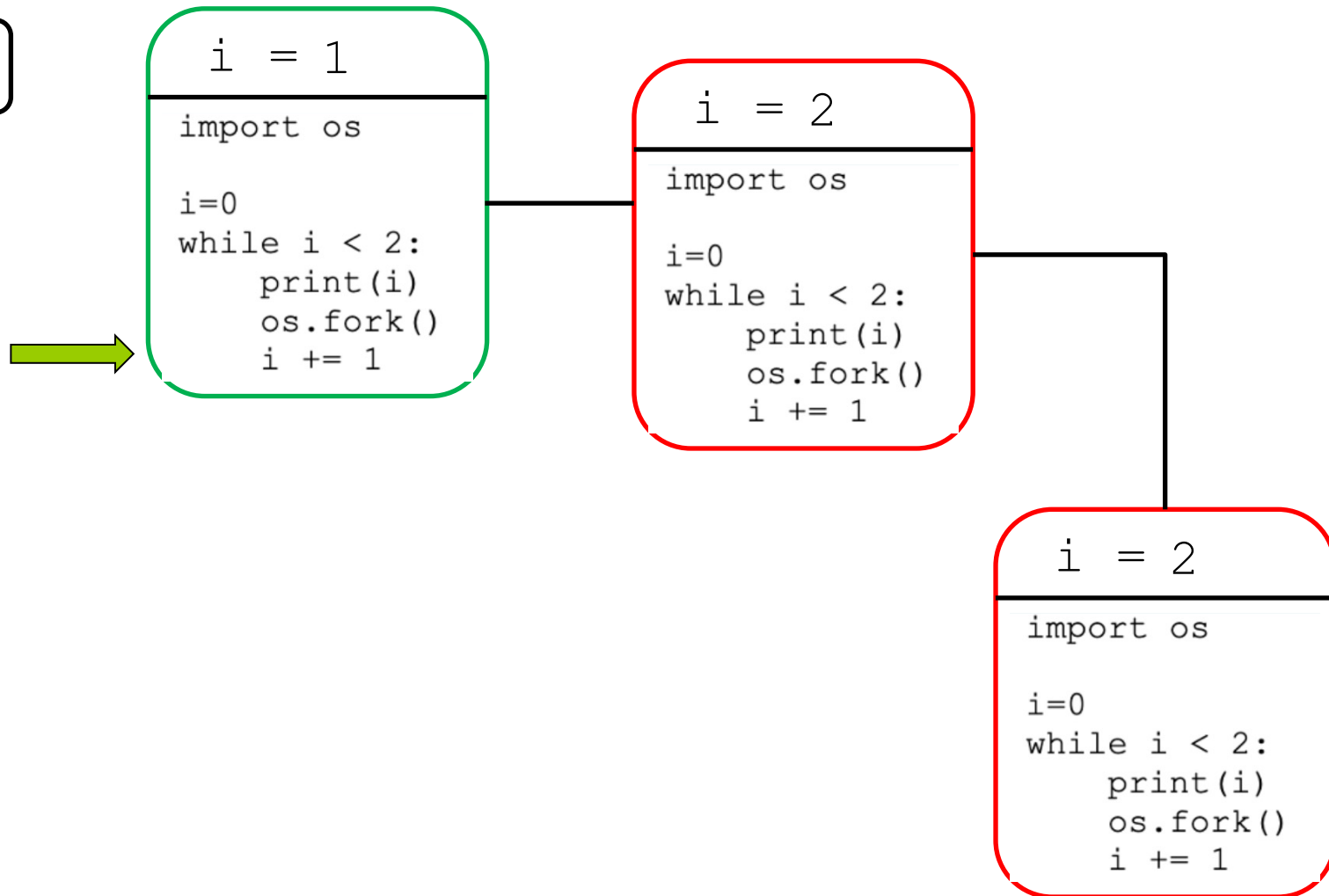
output



# Fork Example

01

output

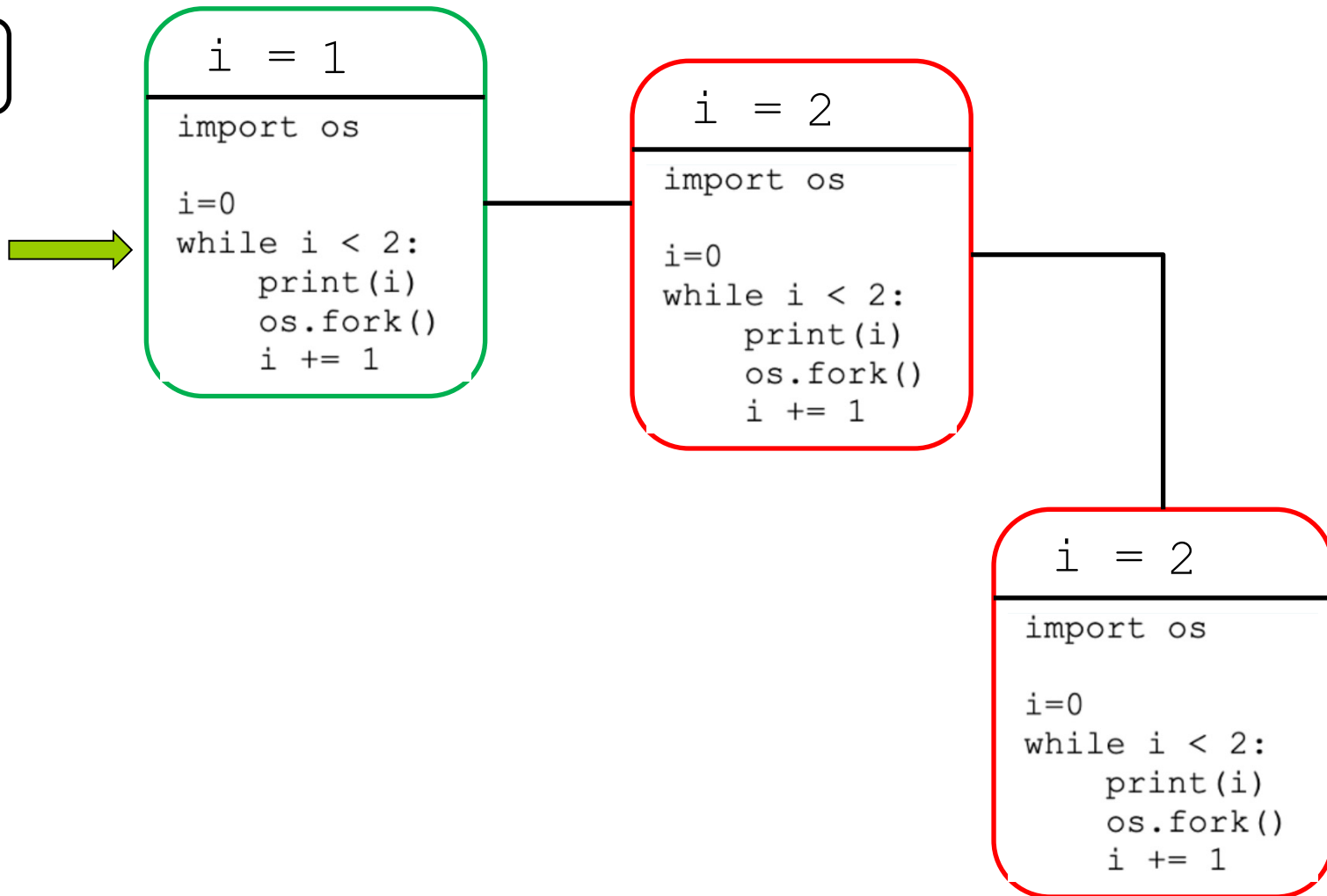




# Fork Example

01

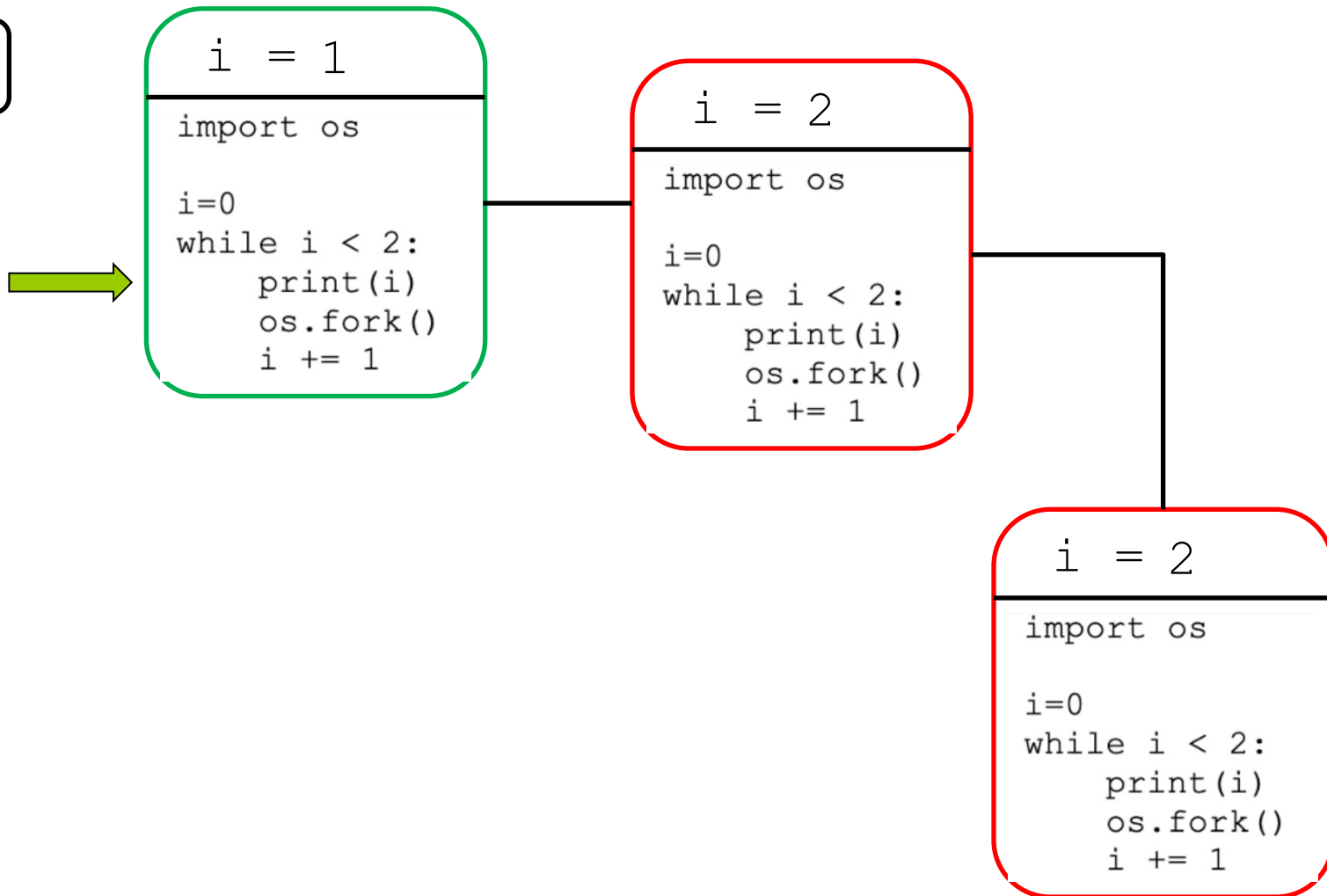
output



# Fork Example

011

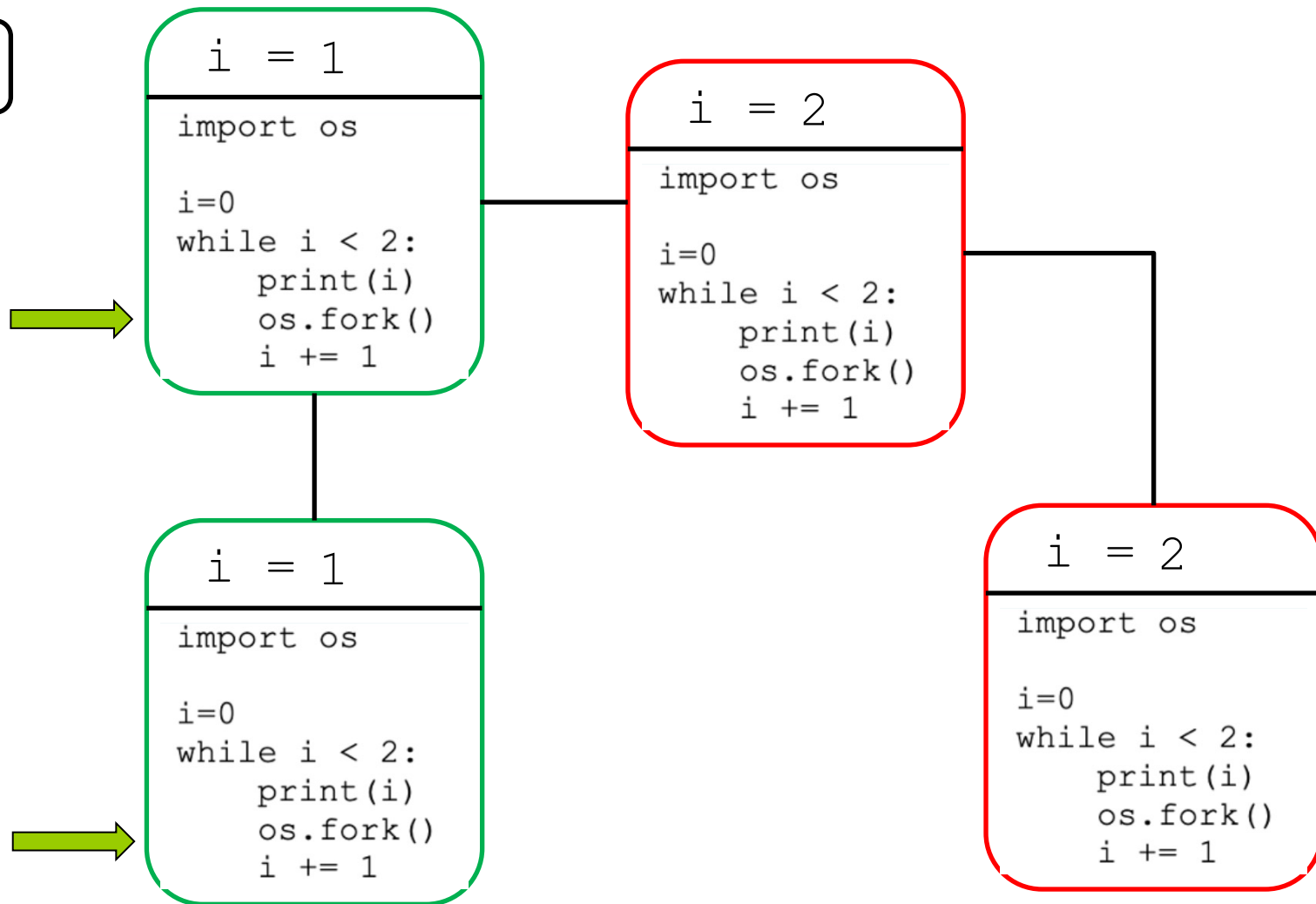
output



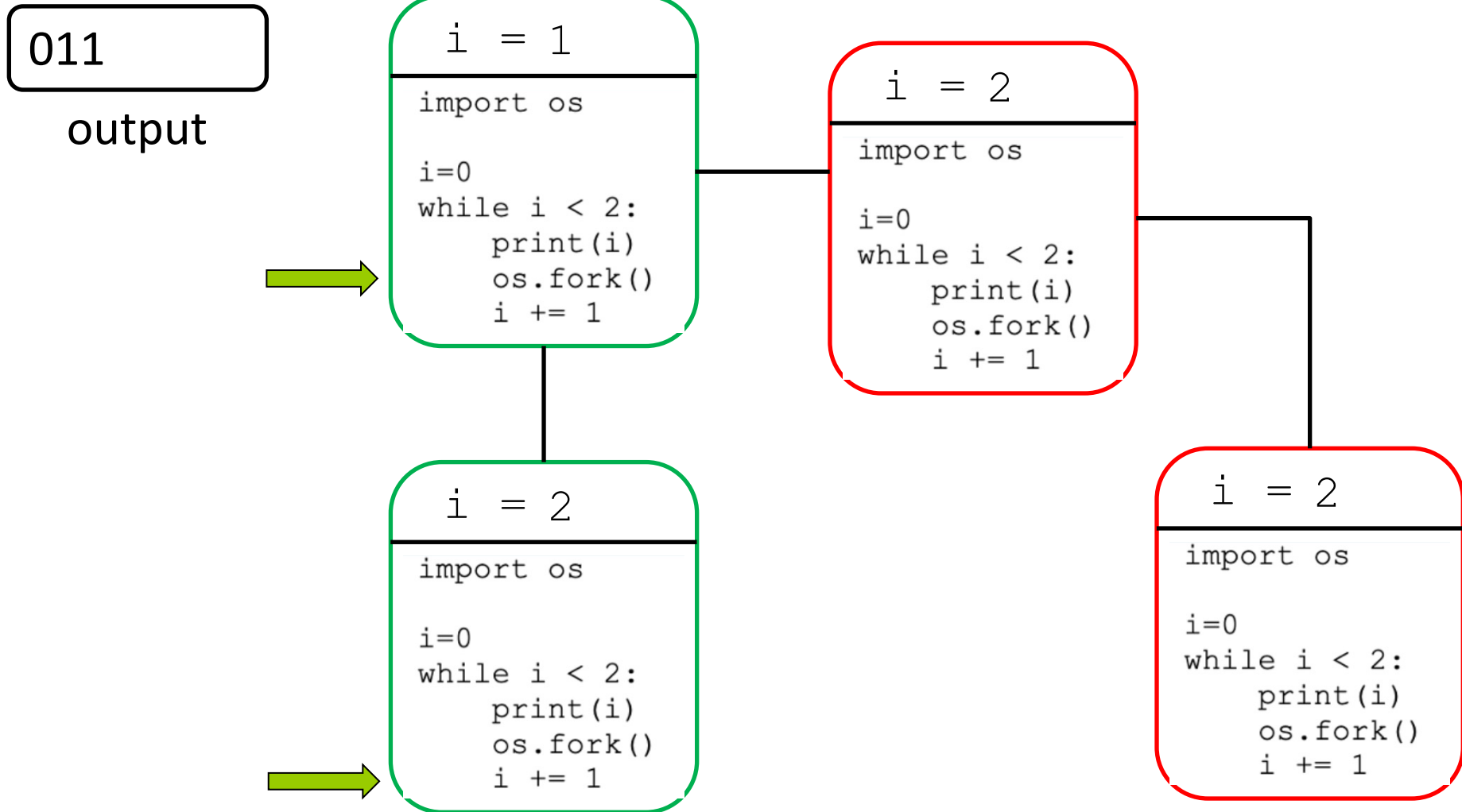
# Fork Example

011

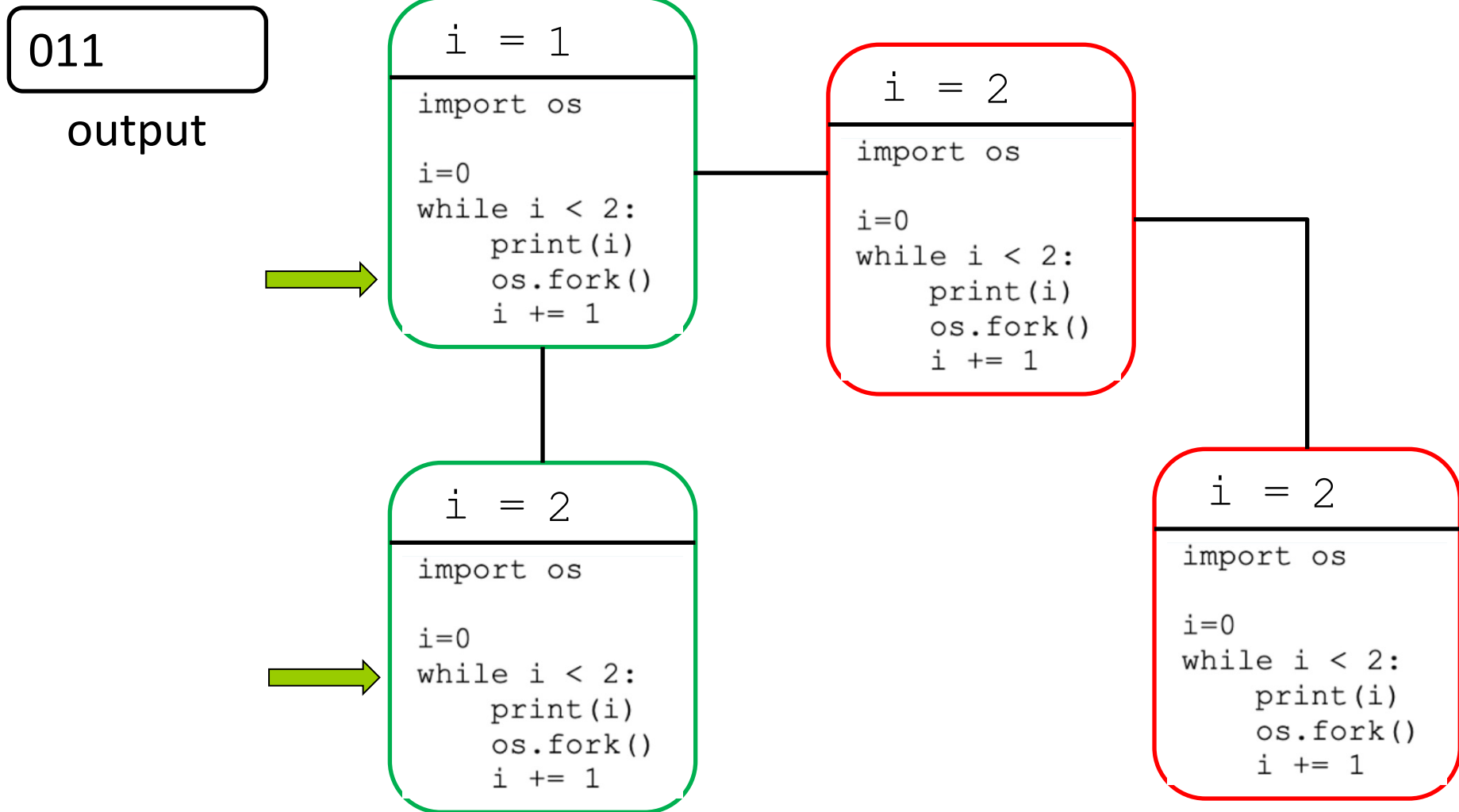
output



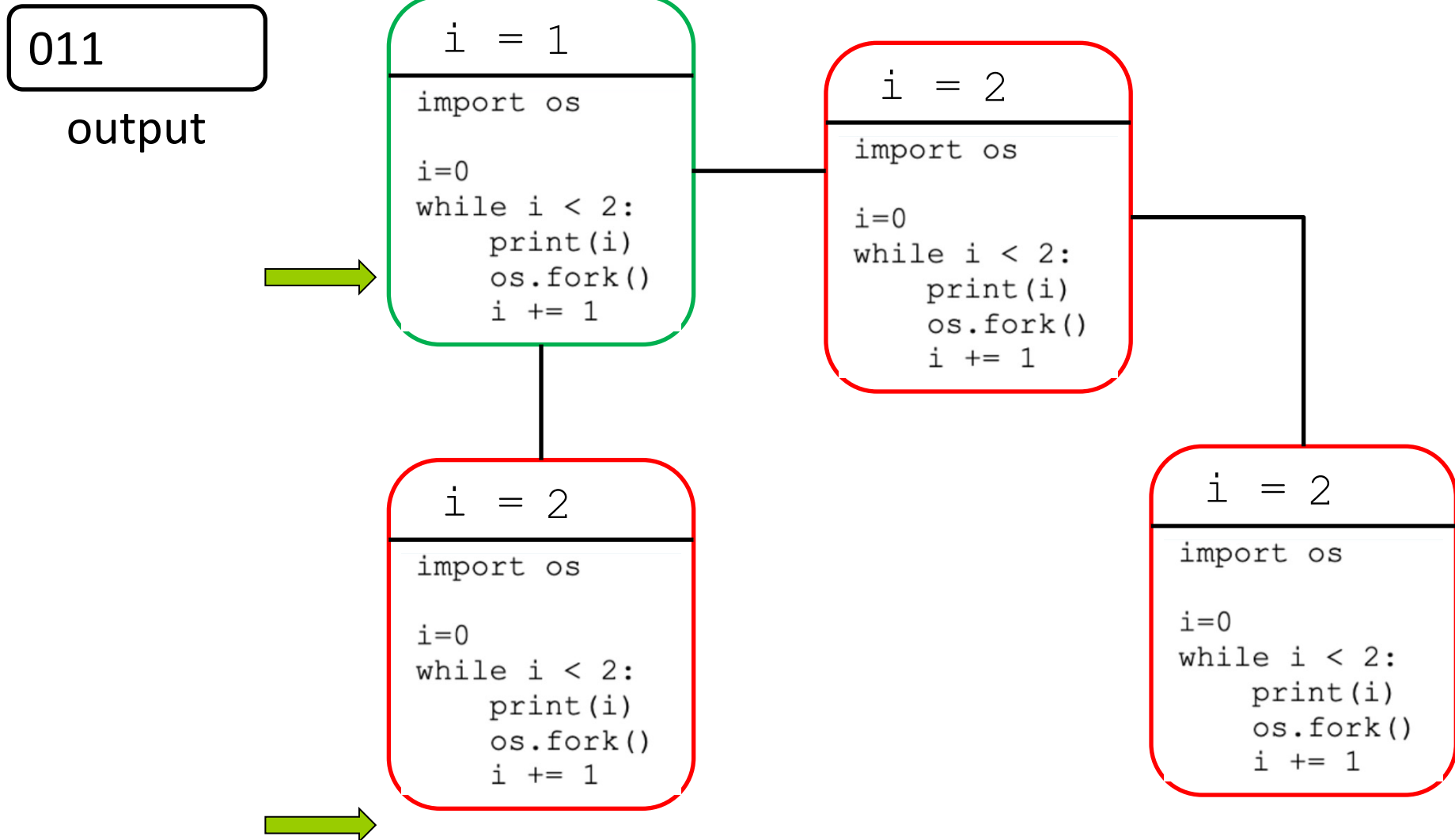
# Fork Example



# Fork Example



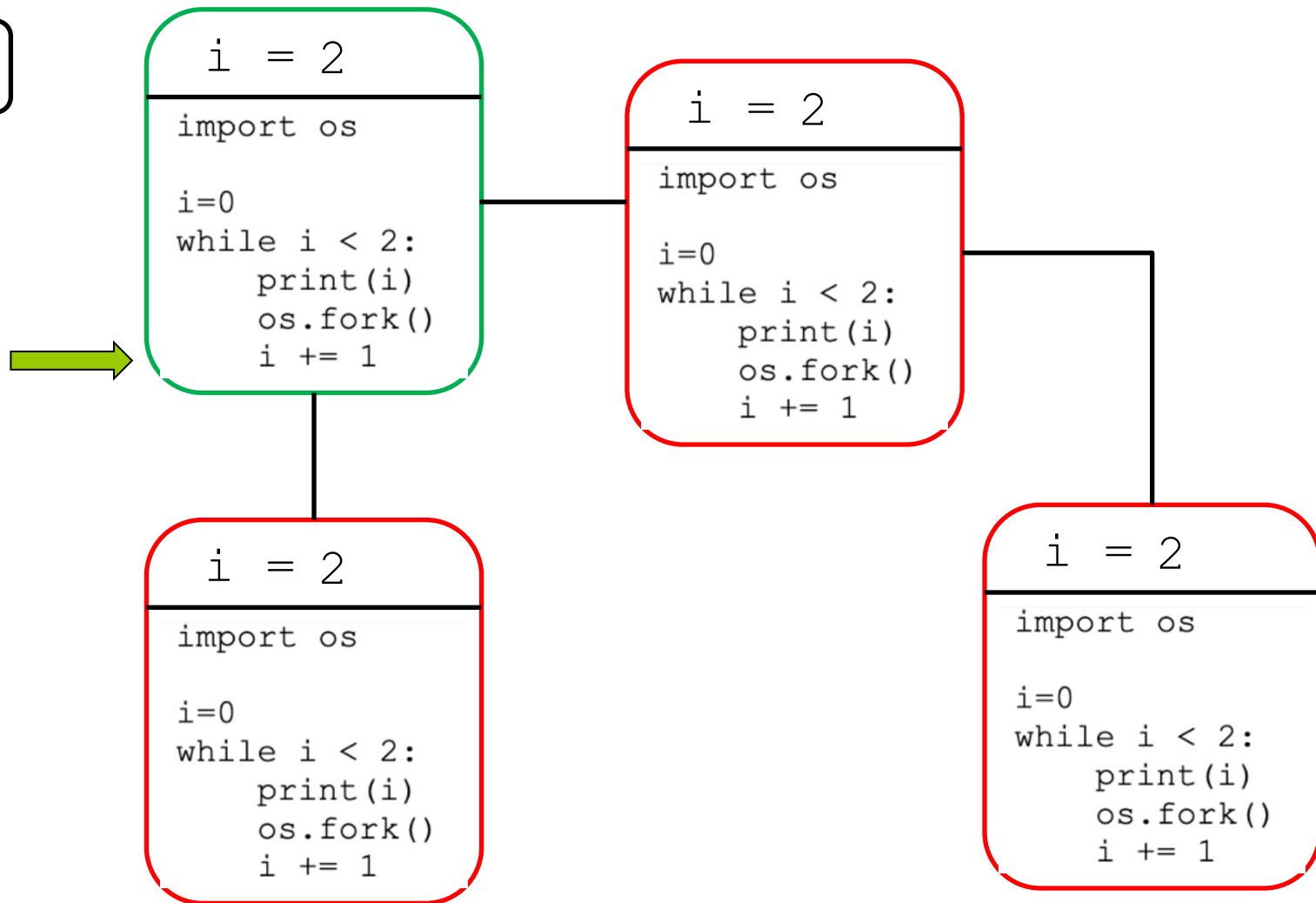
# Fork Example



# Fork Example

011

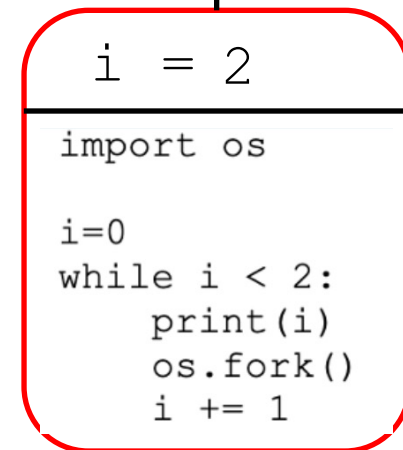
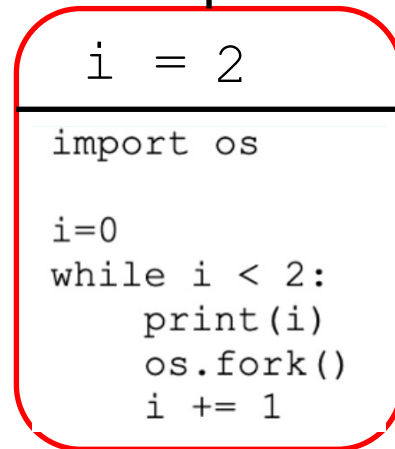
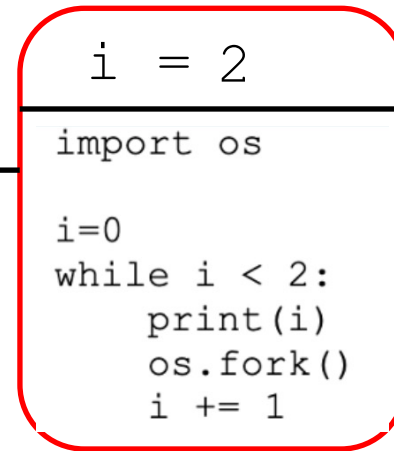
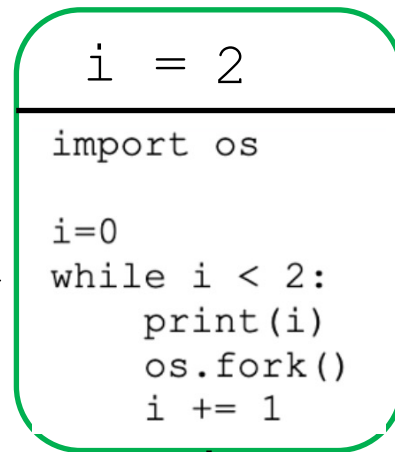
output



# Fork Example

011

output

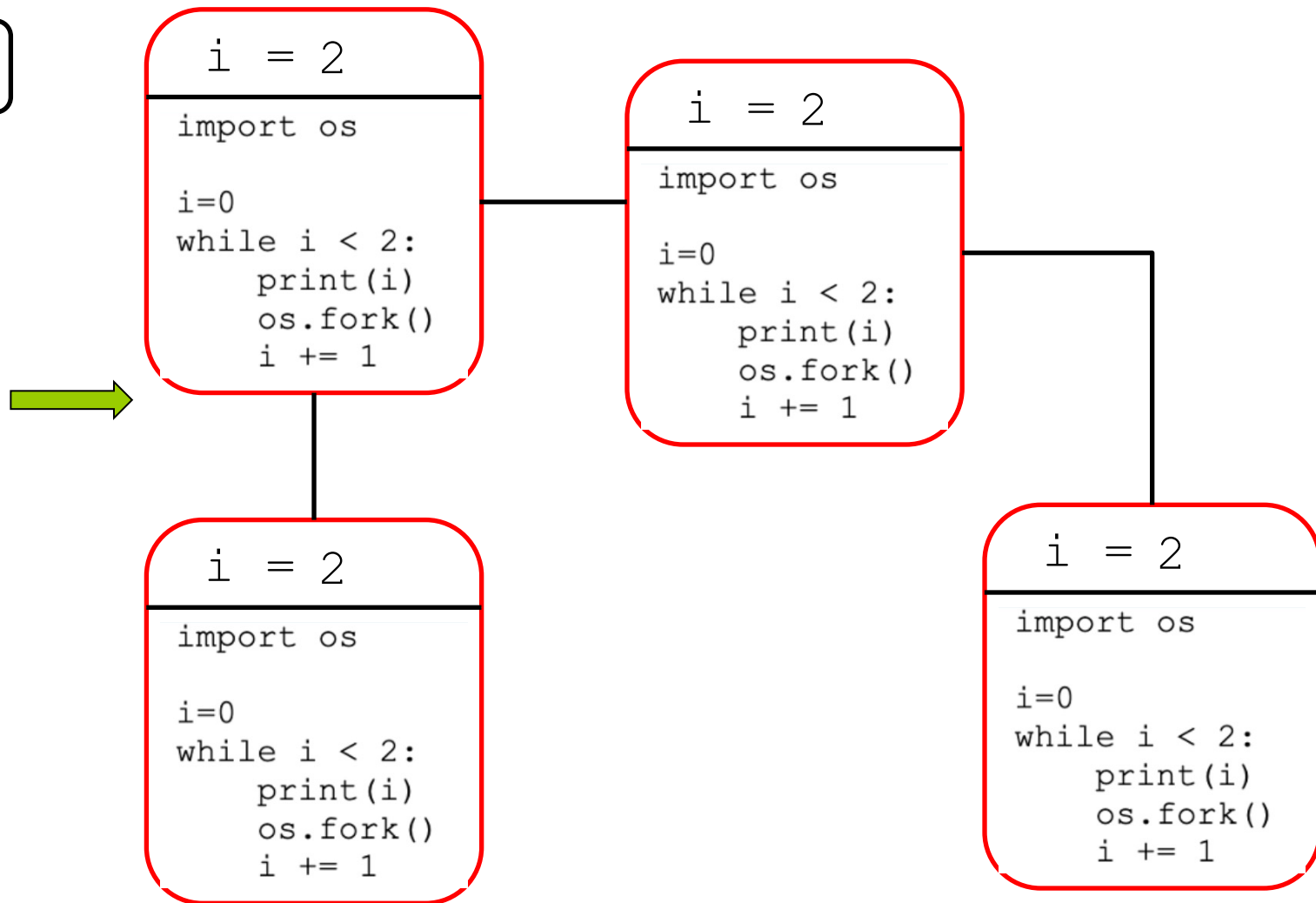




# Fork Example

011

output



# Sample exam question

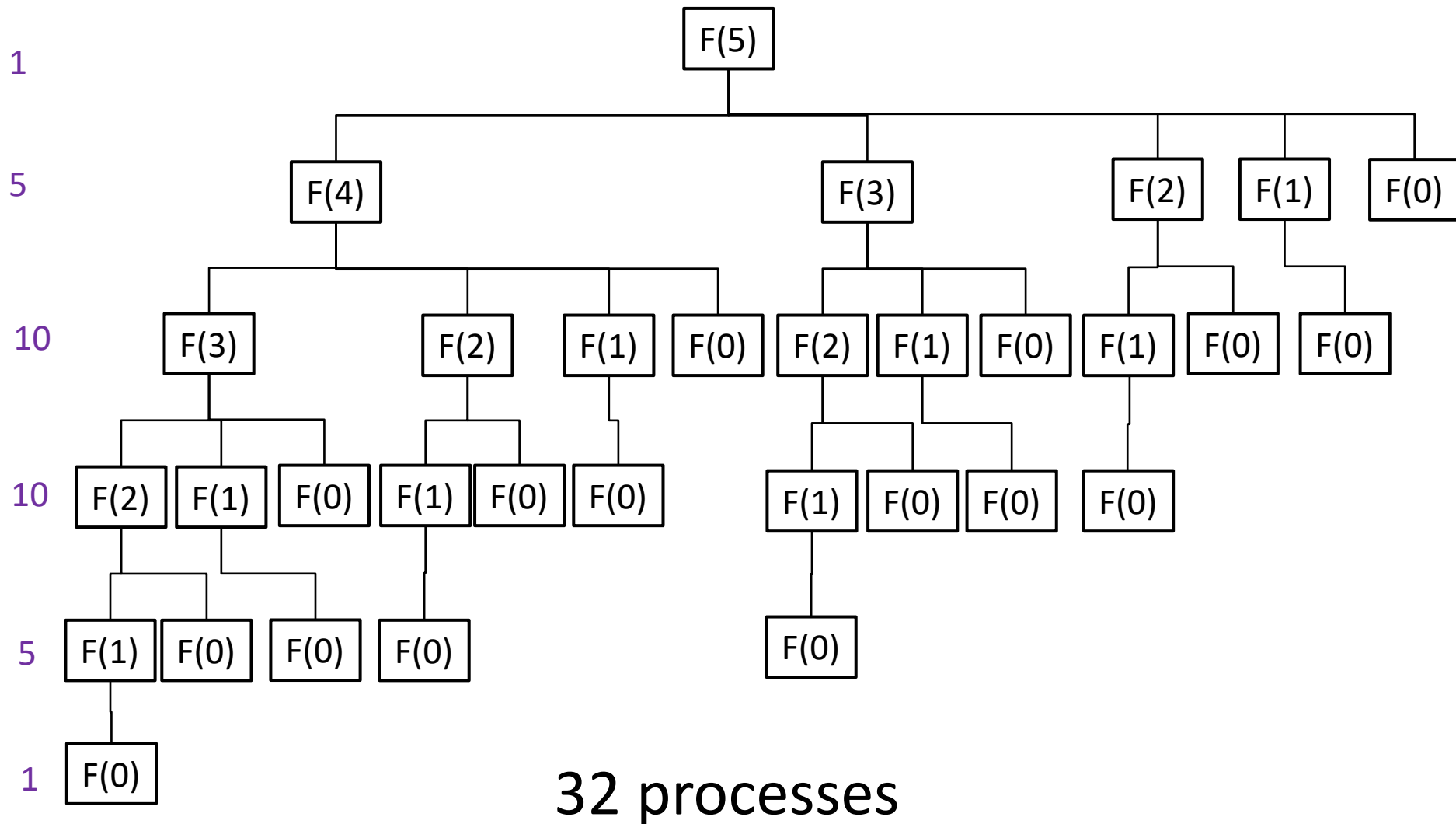
---

۱. در اثر اجرای برنامه زیر چند پردازش ایجاد می شوند. روش محاسبه شما بایستی مشخص باشد و تنها یک عدد برای پاسخ کفایت نمی کند.

```
main() {  
    forkthem(5);  
}  
void forkthem(int n) {  
    if( n > 0 ) {  
        fork();  
        forkthem(n-1);  
    }  
}
```



# Process Tree



# Sample exam question (cont.)

---

۲. خروجی برنامه‌های زیر چیست؟ به شکل خلاصه توضیح دهید.

```
// Program 1
main() {
    int val = 5;
    int pid;
    if (pid = fork())
        wait(pid);

    val++;
    printf ("%d\n", val);
    return val;
}
```



# Sample exam question (cont.)

---

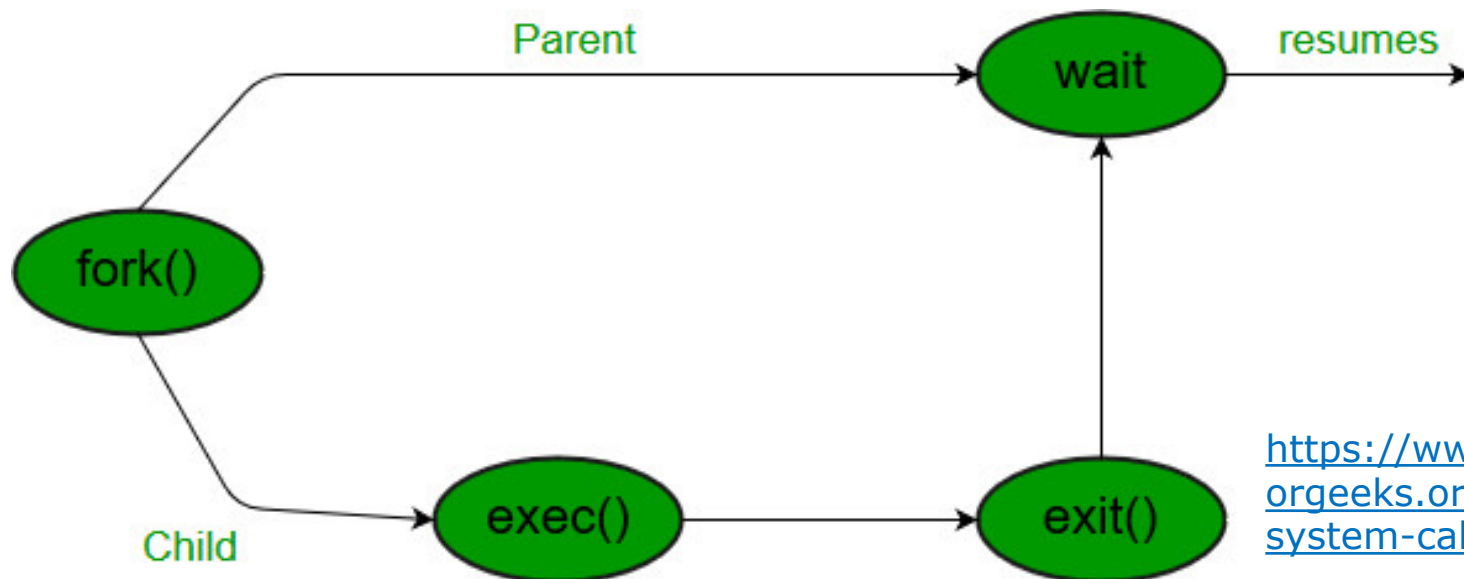
۲. خروجی برنامه‌های زیر چیست؟ به شکل خلاصه توضیح دهید.

```
// Program 2
main() {
    int val = 5;
    int pid;
    if (pid = fork())
        wait(pid);
    else
        exit(val);
    val++;
    printf ("%d\n", val);
    return val;
}
```



# Process Termination

- Process executes last statement and then asks the operating system to ***delete it*** using the **`exit()`** system call.
  - Returns status data from child to parent (via **`wait()`**)
  - Process' resources are deallocated by operating system.



<https://www.geeksforgoeks.org/wait-system-call-c/>

# Process Termination (cont.)

---

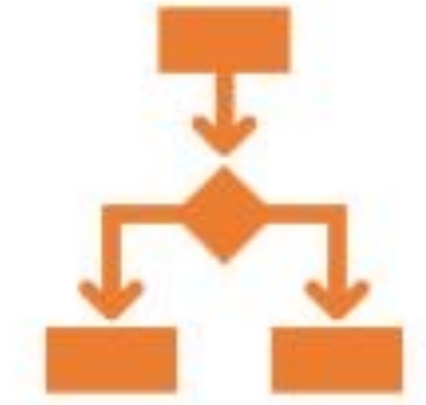
- Parent may terminate the execution of children processes using the `abort()` system call.
  
- Some reasons for doing so:
  - Child has exceeded allocated resources.
  - Task assigned to child is no longer required.
  - The parent is exiting, and the operating systems does not allow a child to continue if its parent terminates.



# Process Termination (cont.)

---

- Some OSs do not allow child to ***exists*** if its parent has terminated.
  - If a process terminates, then all its children must also be terminated.
  - **Cascading termination:** All children, grandchildren, etc., are terminated.
  - The termination is initiated by the operating system.





# Process Termination (cont.)

---

- The parent process may wait for termination of a child process by using the ***wait()*** system call.
  - The call returns status information and the pid of the terminated process.

***pid = wait(&status);***

- If no parent waiting (did not invoke wait()), process is a **zombie**.
- If parent terminated without invoking wait(), process is an **orphan**.

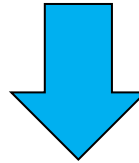


# Multiprocess Architecture – Browser

---

- Many web browsers ran as single process (some still do)

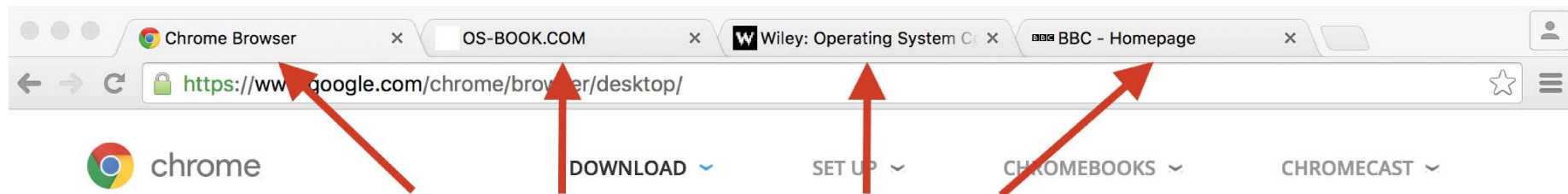
If one web site causes trouble



Entire browser can hang or crash

# Multiprocess Architecture – Chrome Browser (cont.)

- Google Chrome is multiprocess with 3 different types of processes:
  - **Browser** process manages user interface, disk and network I/O.
  - **Renderer** process renders web pages, deals with HTML, Javascript.
    - ▶ A new renderer created for each website opened
    - ▶ Runs in **sandbox** restricting disk and network I/O (why?)
  - **Plug-in** process for each type of plug-in.



Each tab represents a separate process.