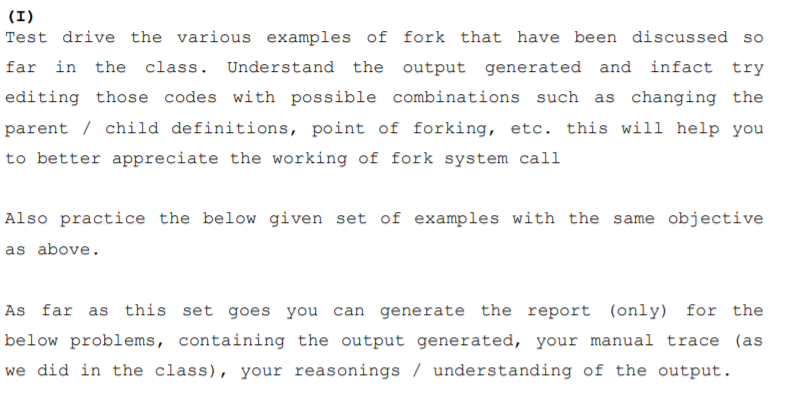
**OPERATING SYSTEMS LAB ASSIGNMENT 2**

T KARTHIKEYAN

CED18I064

**QUESTION 1**

****

**HAND TRACE**

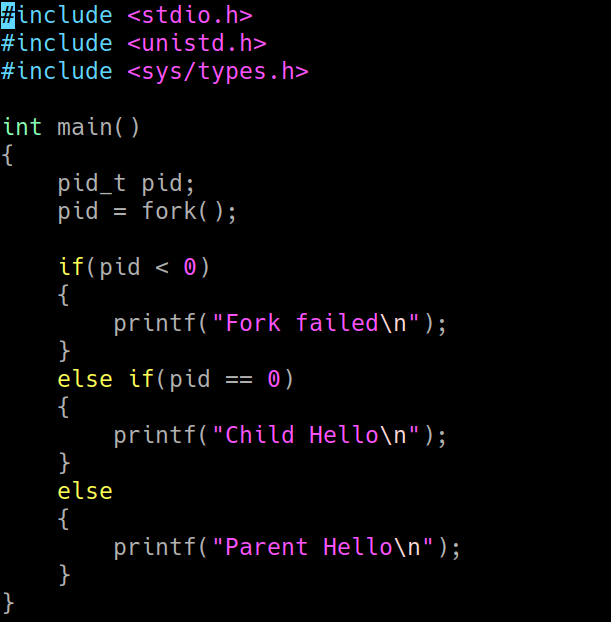
During fork(), the parent process creates another duplicate process which is named child process which does the same job as parent process after the point of forking.

Expected output :

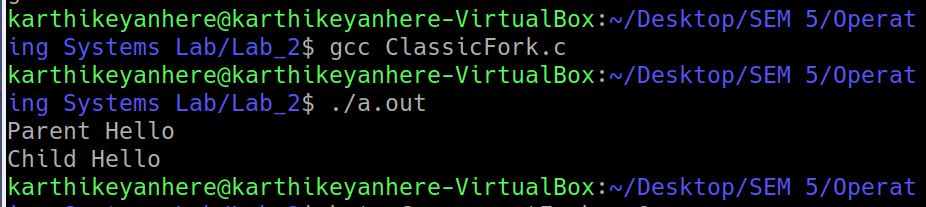
Child Hello (or Parent Hello)

Parent Hello (or Child Hello)

**CODE**

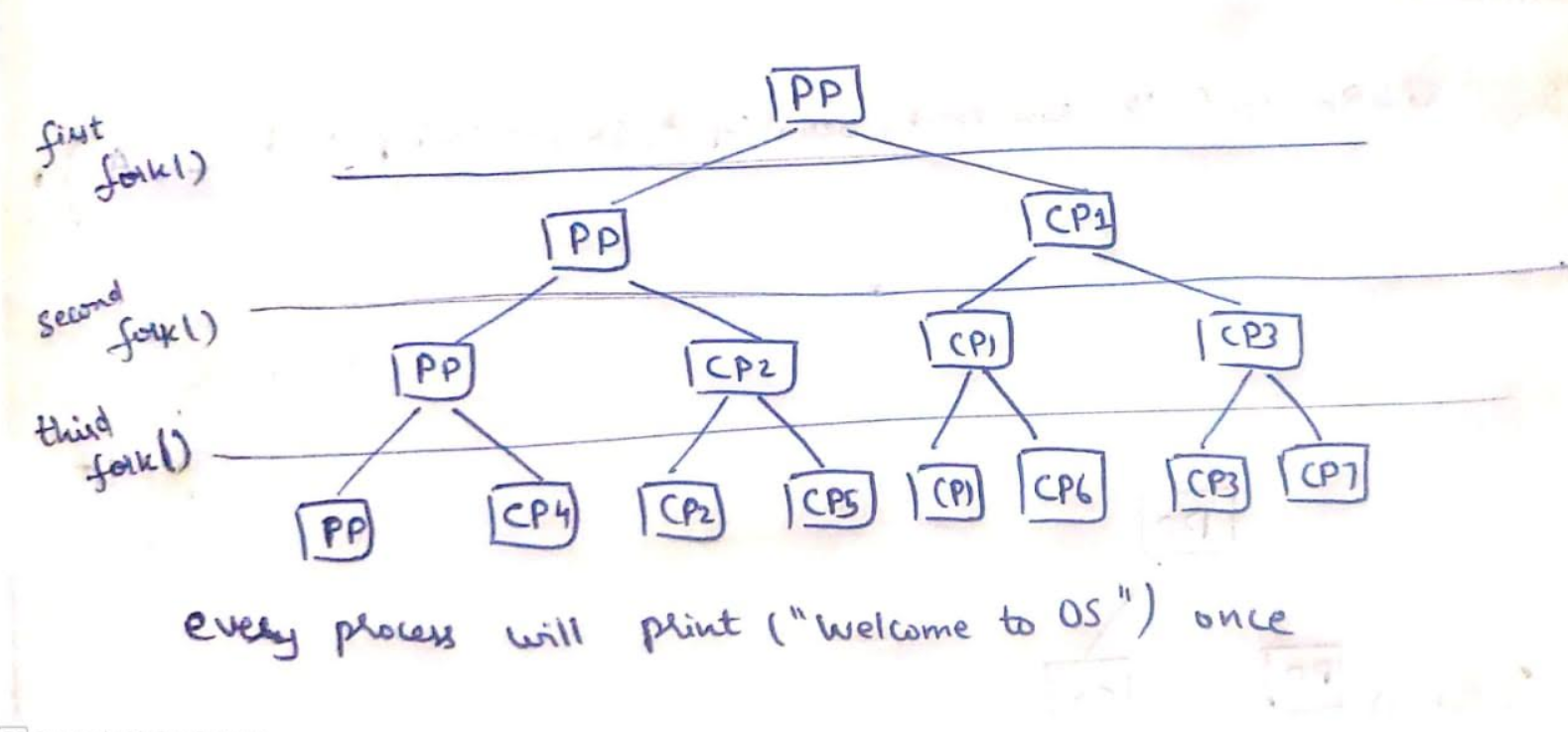
****

**OUTPUT**

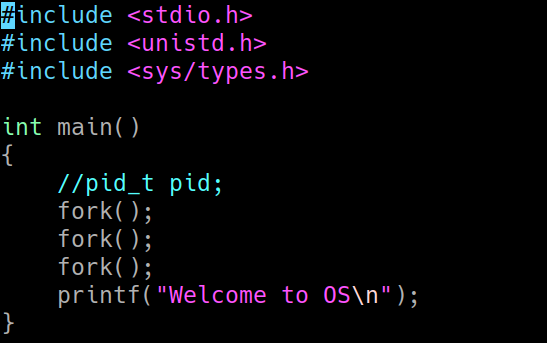
****

**HAND TRACE**

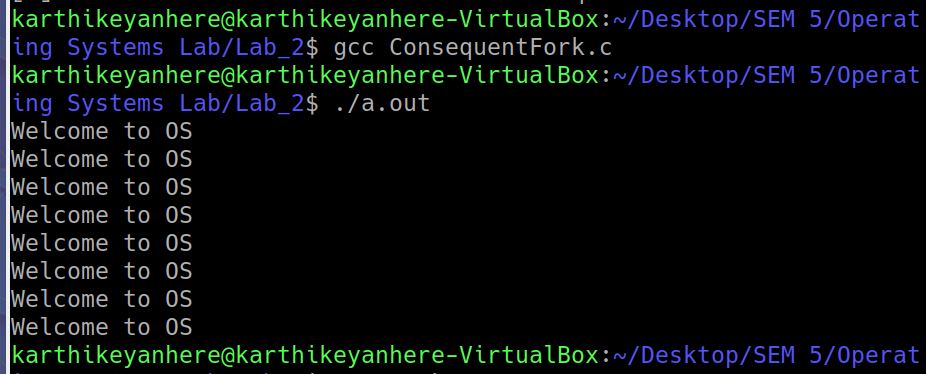
n consecutive fork() creates 2^n - 1 child processes as described below, each child process along with parent process prints (“Welcome to OS”) once

****

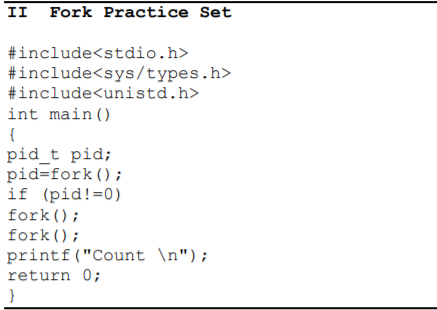
**CODE**

****

**OUTPUT**

****

**QUESTION 2**

****

**HAND TRACE**

During the first fork, the parent process has two consecutive processes resulting in three child processes while the child process(during first fork) has one fork resulting in one more additional child.

Total = parent + 1 + 3 + 1 = 6.

Each process accounting for one “Count”

Expected output :

Count

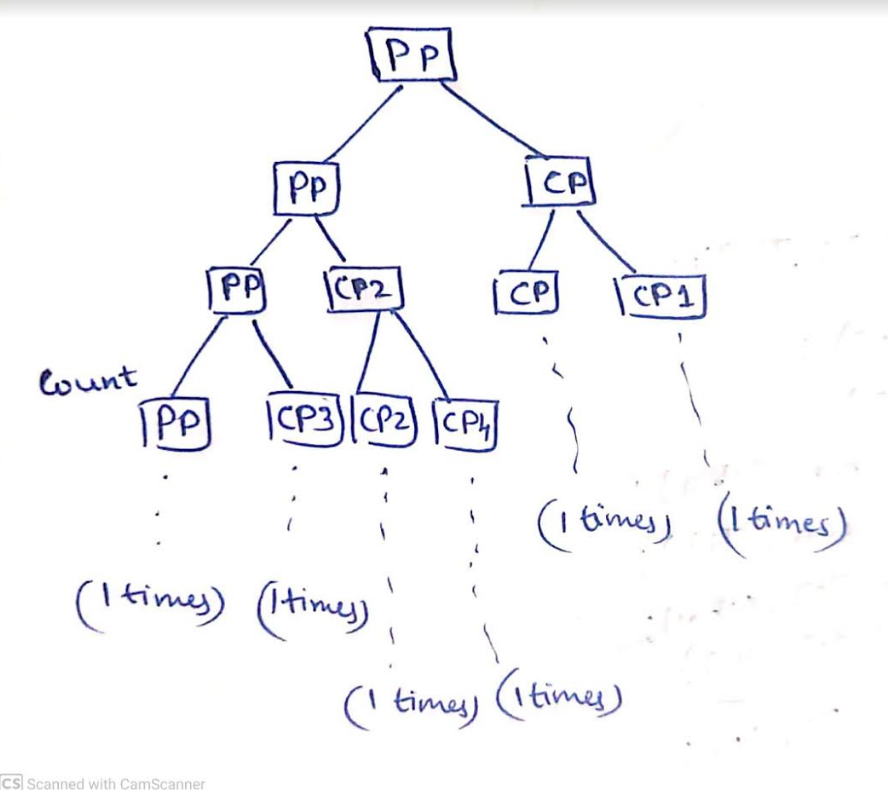
Count

Count

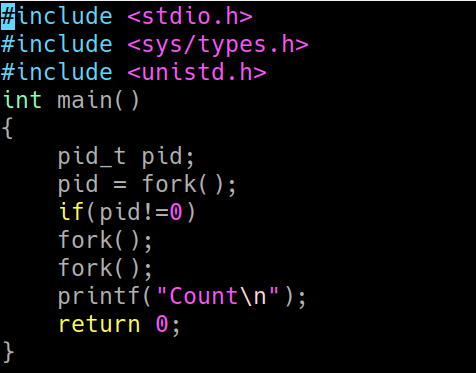
Count

Count

Count

****

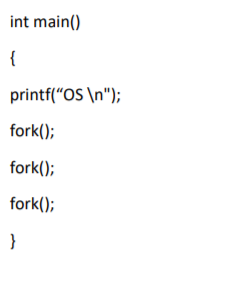
**CODE**

****

**OUTPUT**

****

**QUESTION 3**

****

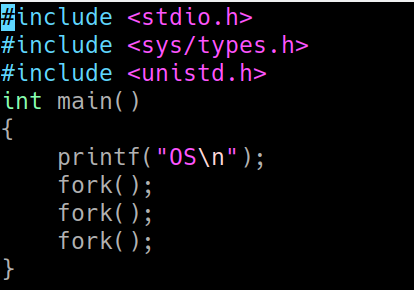
**HAND TRACE**

Expected output:

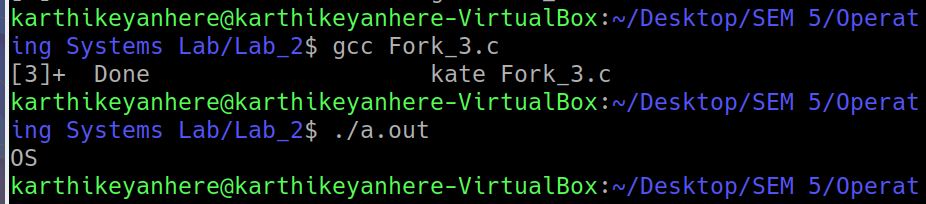
OS (main function)

Although 7 child process will be created but no alteration in output

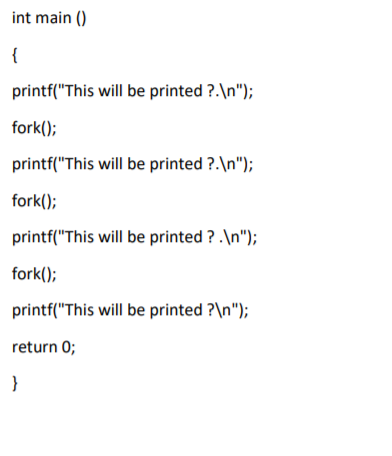
**CODE**

****

**OUTPUT**

****

**QUESTION 4**

****

**HAND TRACE**

During first fork(), child process CP1 is created

Parent process(PP) has 4 print statements

CP1 has 3 print statements

During second fork(), both PP and CP1 creates CP2 and CP4 respectively

Both CP2 and CP4 got 2 print statements each

During third fork(), PP , CP2 , CP1 and CP4 creates CP5 , CP7 , CP3 and CP6 respectively

CP5 , CP7 , CP3 and CP6 got 1 print statements each

Total = 4 + 3 + 2 + 2 + 1 + 1 + 1 + 1 = 15 print statements

Expected output:

This will be printed?

This will be printed?

This will be printed?

This will be printed?

This will be printed?

This will be printed?

This will be printed?

This will be printed?

This will be printed?

This will be printed?

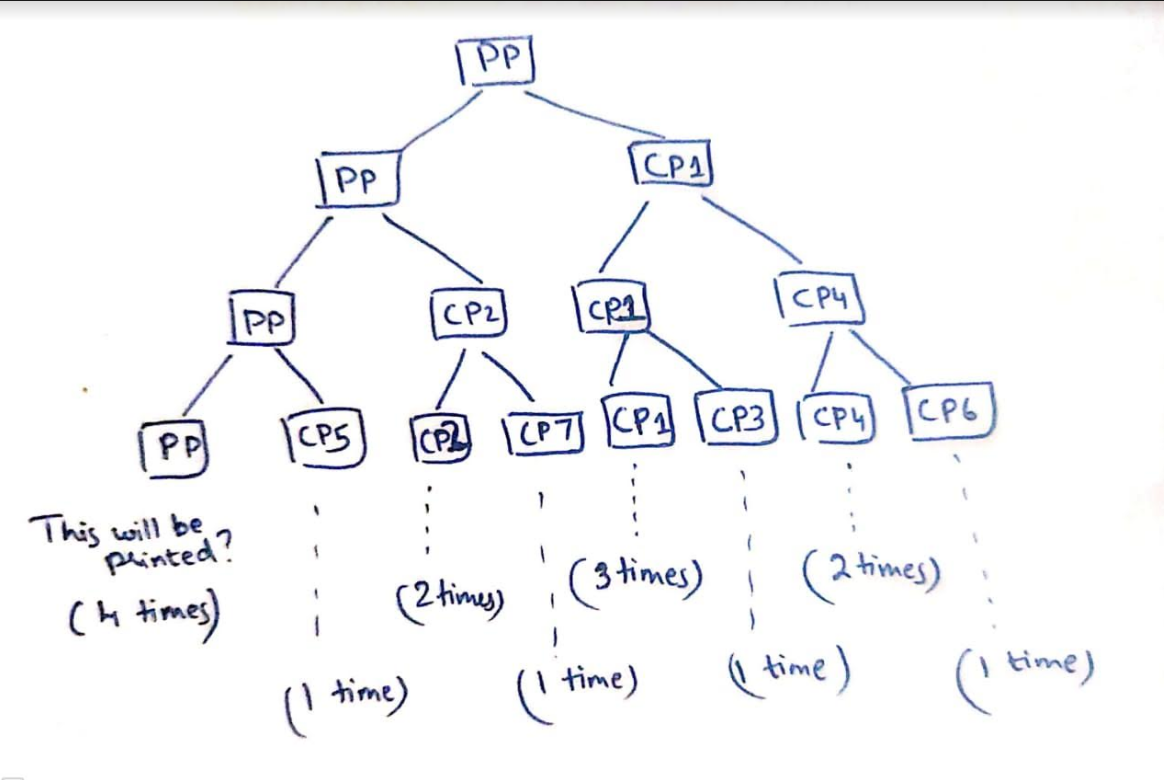
This will be printed?

This will be printed?

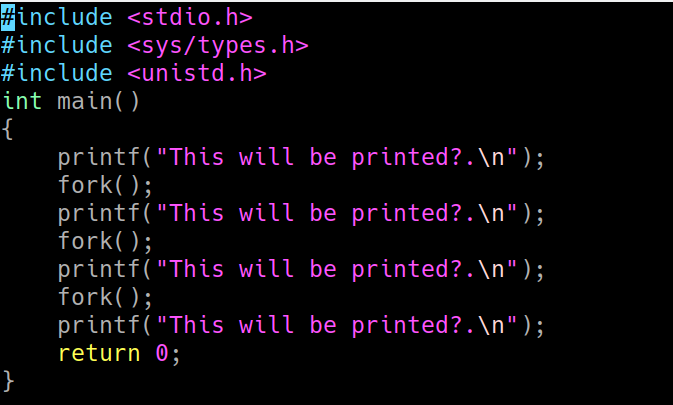
This will be printed?

This will be printed?

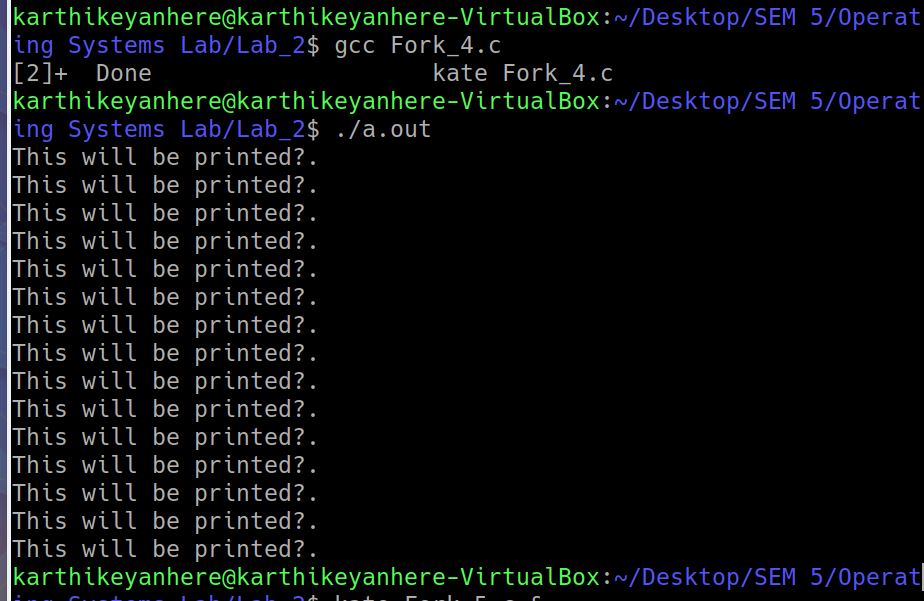
This will be printed?



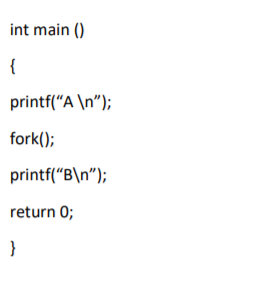
**CODE**

****

**OUTPUT**

****

**QUESTION 5**

****

**HAND TRACE**

A is printed initially,

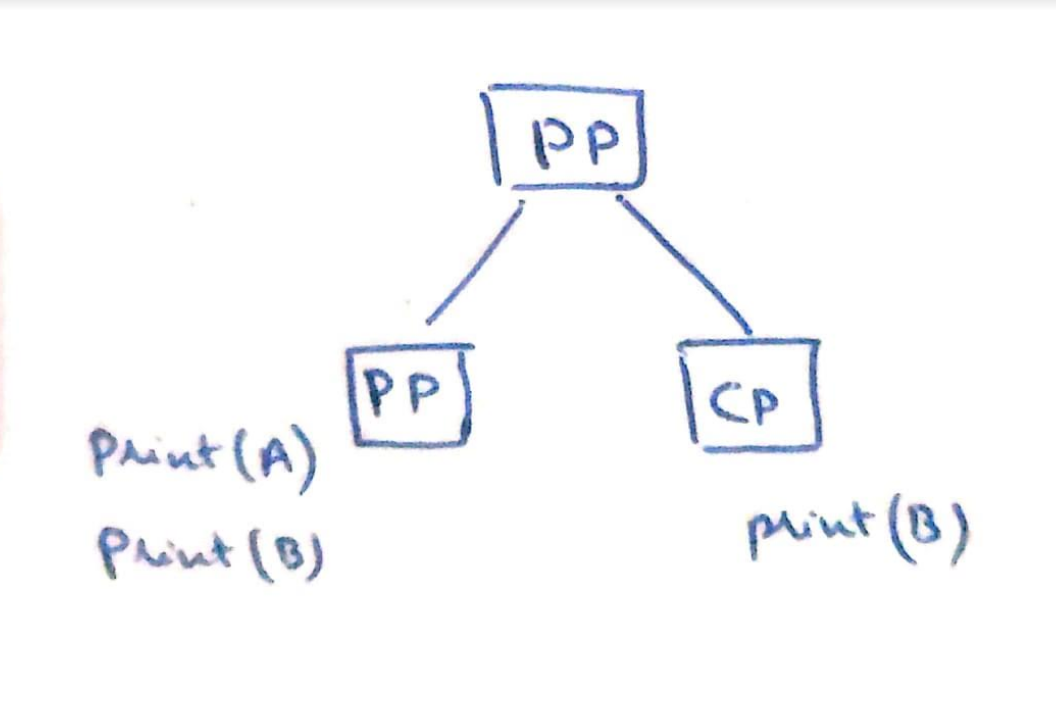
During the fork() call, parent remains as it is while child is created with the statements after printf(“B”)

Expected output :

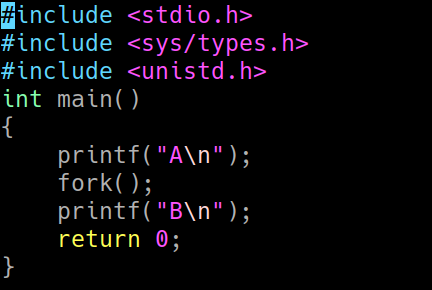
A (main function)

B (due to parent)

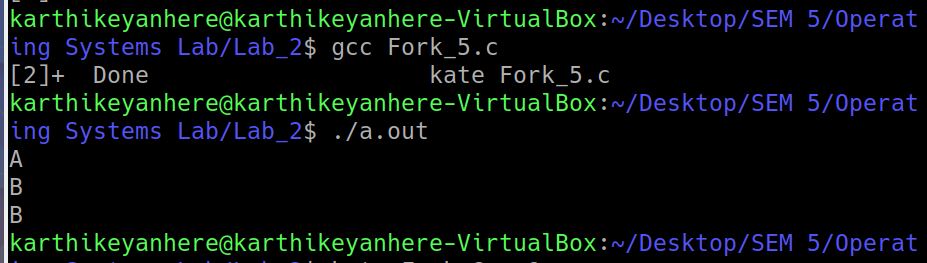
B (due to child)



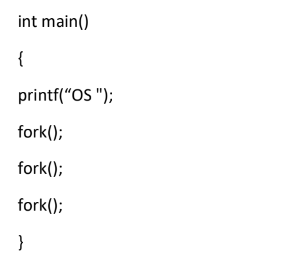
**CODE**

****

**OUTPUT**

****

**QUESTION 6**

****

**HAND TRACE**

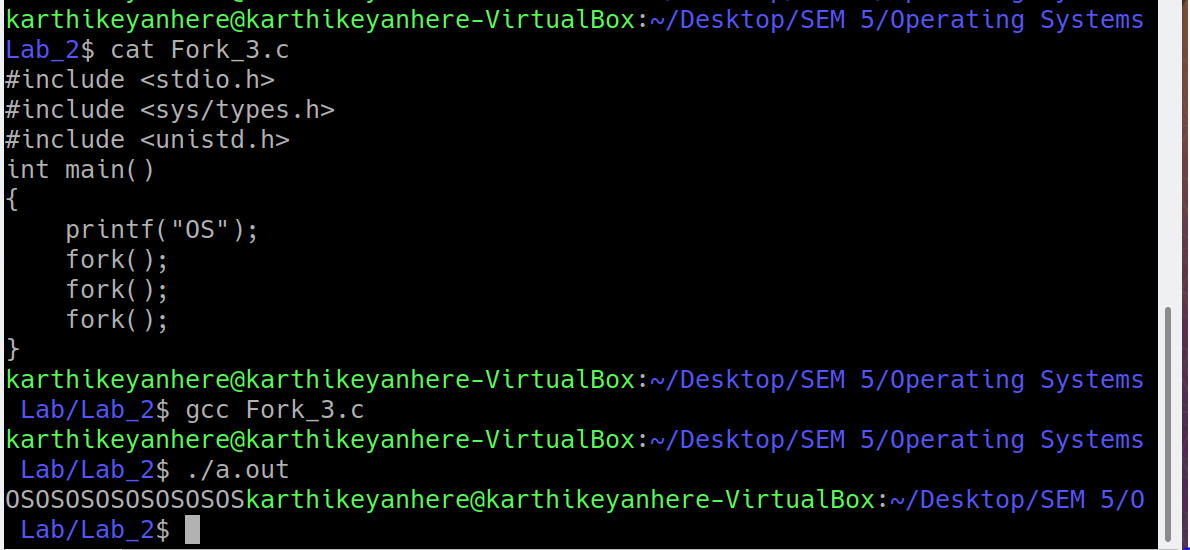
printf() buffers output until a newline is encountered. So this version inserts “OS” into the output buffer, and then forks.

Since all the seven child processes created as part of three consecutive forks will have an output buffer that contains “OS”

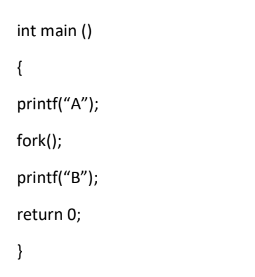
Expected output :

OSOSOSOSOSOSOSOS

**CODE AND OUTPUT**

****

**QUESTION 7**

****

**HAND TRACE**

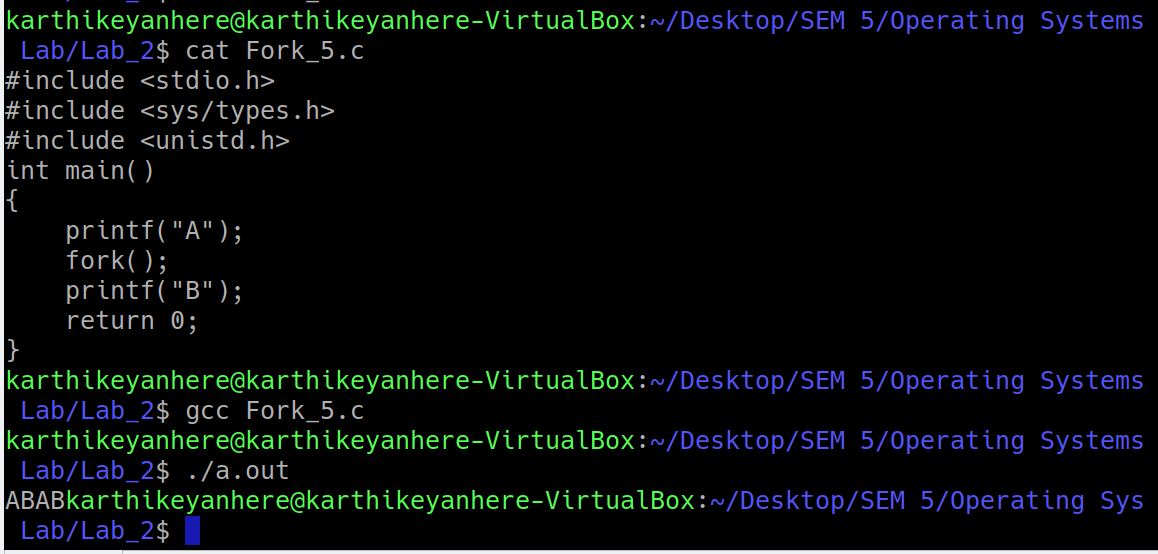
printf() buffers output until a newline is encountered. So this version inserts A into the output buffer, and then forks.

Since both processes are identical, they now both have an output buffer that contains A.

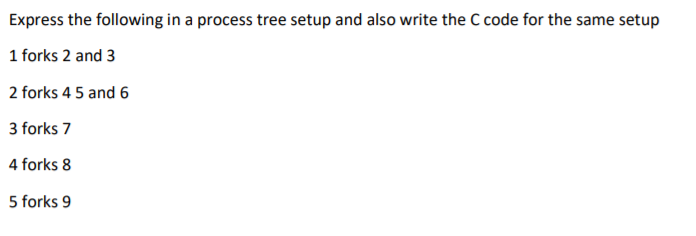
Expected output:

ABAB (printing order may be different)

**CODE AND OUTPUT**

****

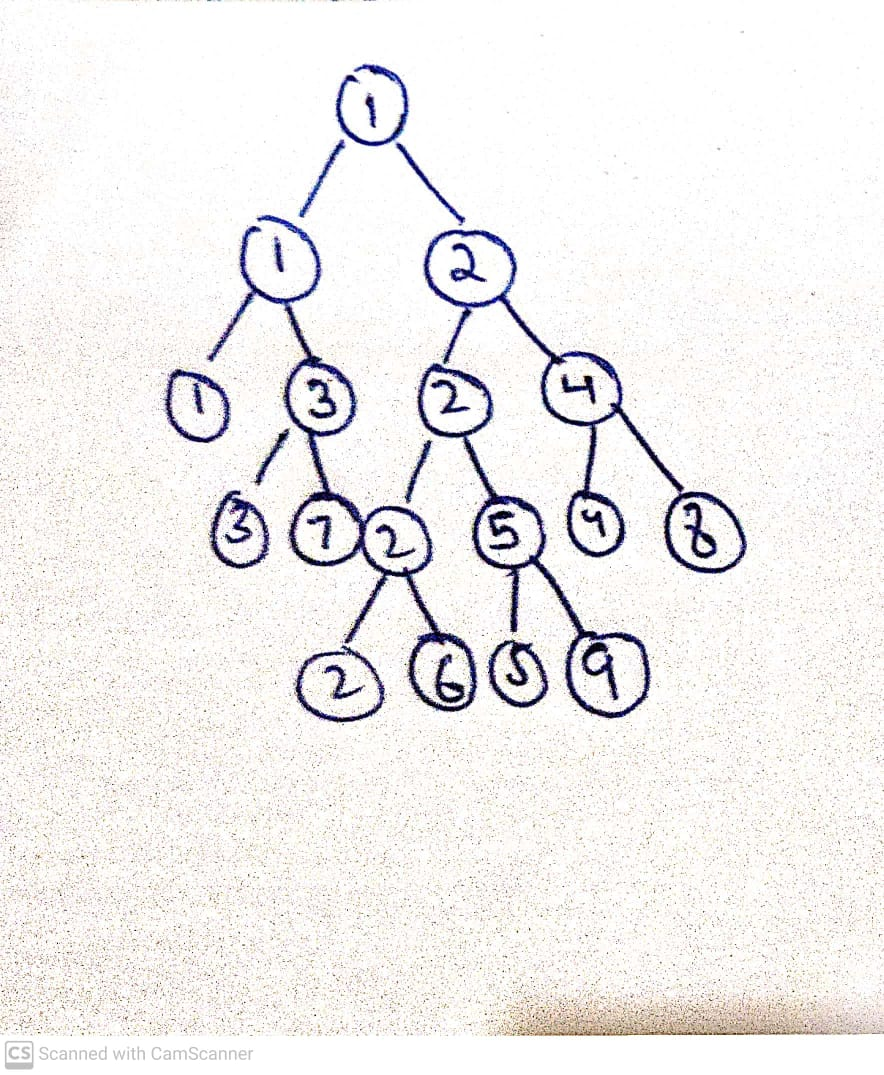
**QUESTION 8**

****

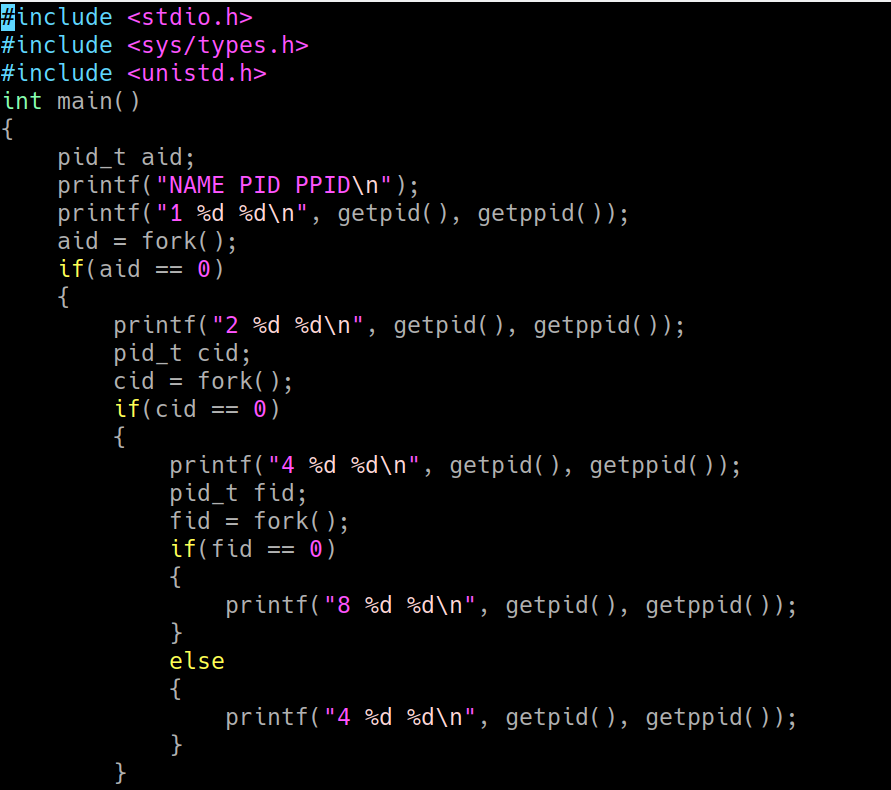
**HAND TRACE**

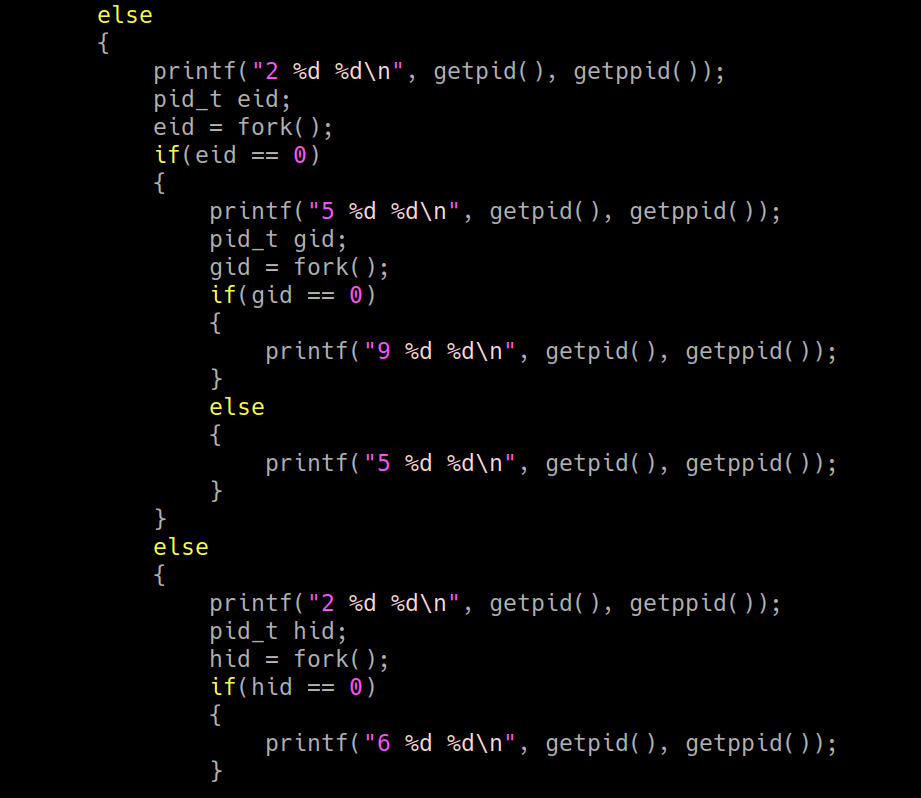
1 forks 2 and 3 is realized using two forks, during first fork(), 1 creates 2 followed by 3 during second fork() …

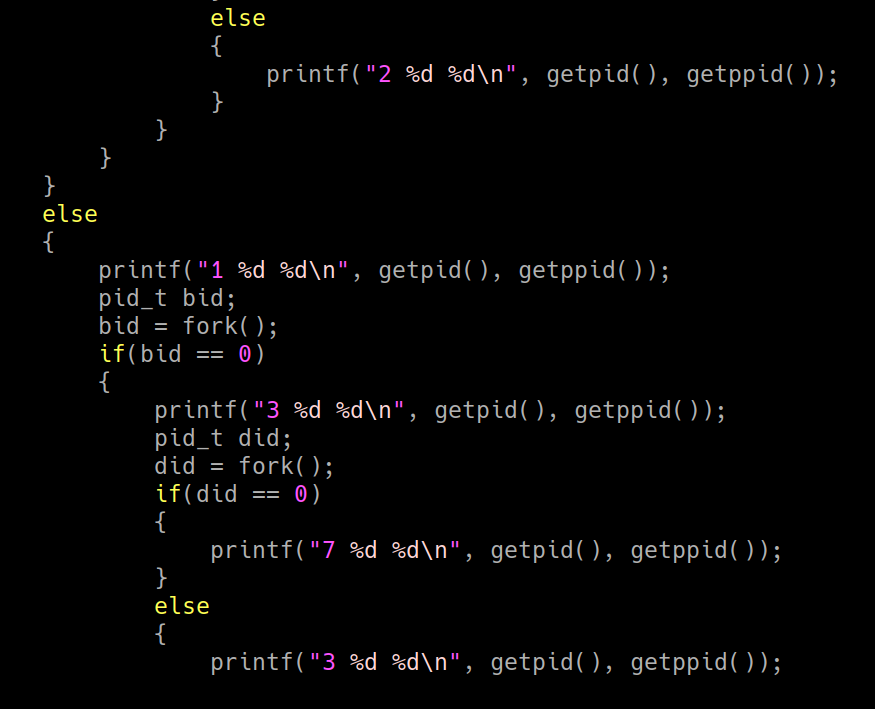
Similarly other forks are realized

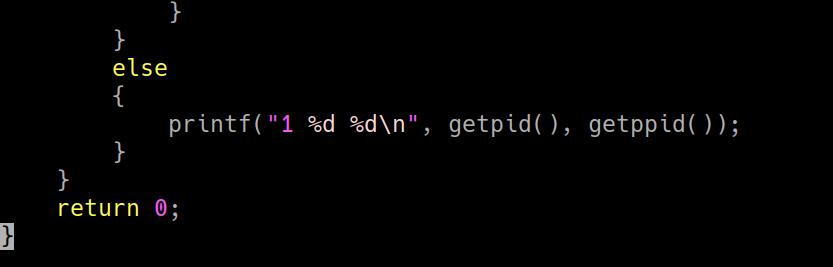
****

**CODE**

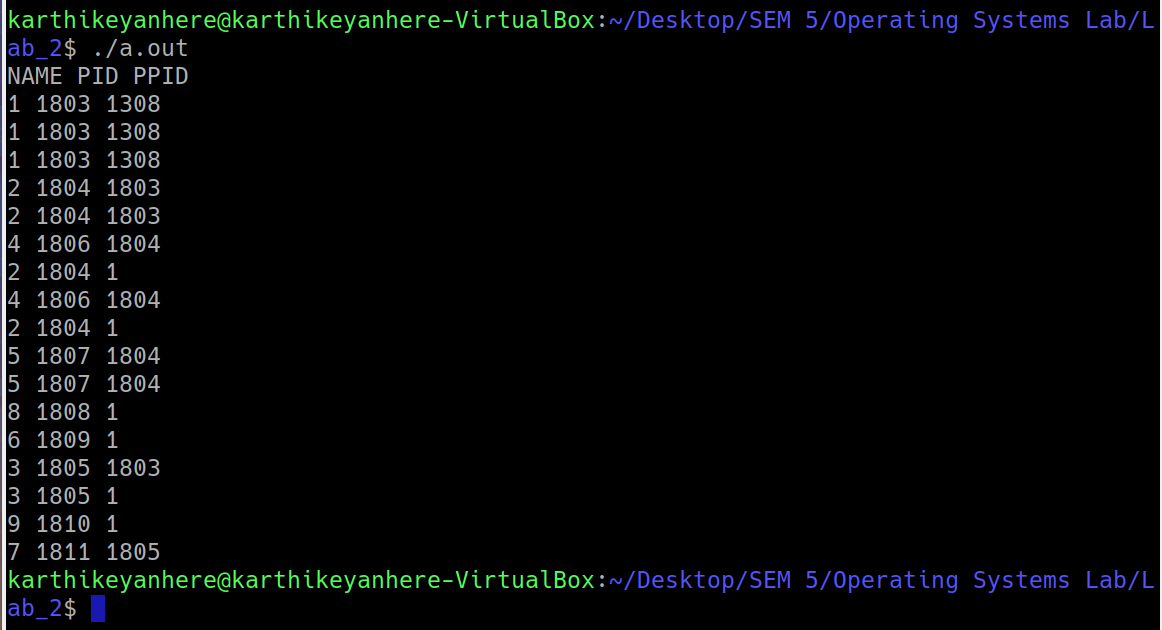
****

****

****

****

**OUTPUT**

****