Name: Kv Sridhar Sai
Tutorial-1
Roll: CB.EN.U4CSE19063 Operating Systems

(1.)

```
# include <stdio.h>
   int main() {
       int a = 10;
       if (fork() = =0) {
            a = a + 5;
            printf("%d \n", a);
       }
       else {
            wait();
            a = a - 5;
            printf("%d \n", a);
       }
   }
```

Sol:- The output of this code would be 15 and 5.

→ Tracing:

Step-1: a = 10

Step-2: fork() = = 0 ⟶ child process is created if the return value of fork is 0;

and in the else part we have the code for the parent process.

So we should move to parent process first. (i.e) The else part

→ Parent process code: (else part:)

```
wait();
a = a - 5;
printf("%d\n", a);
```

Now since there is a wait system call in parent process.

function of wait() system call:

Waits till one of its child process completes.

Now control goes to the child process (i.e)(The if part)

if (fork() == 0) → True (child process)

a = a + 5 → a = 10 initially
printf("%d\n", a)

Now a = 10 + 5
= 15

prints '15'

Now the parent process continues.

a = a - 5; → ~~[illegible]~~ parent & child are 2 different processes (cont once the memory gets copied to child it never gets updated due to parent.

So here also $\boxed{a = 10 \text{ (remains)}}$

parent process continuation:

a = a - 5; ⟶ a = 10 - 5;

printf ("%d \n", a); → prints '5'

ext page)