LAB-06

Exercise:

1) Implement the above code and paste the screen shot of the output.

PROGRAM:

```
#include <stdio.h>
#define n 7
int completedPhilo = 0, i;
struct fork {     int
taken;
} ForkAvail[n];

struct philosopher {
   int left;     int
right;
} PhiloStatus[n];
void goForDinner(int philID) {
   // Case: Philosopher has completed dinner     if (PhiloStatus[philID].left ==
10 && PhiloStatus[philID].right == 10) {        printf("Philosopher %d
completed his dinner\n", philID + 1);
   }
   // Case: Philosopher has taken both forks     else if (PhiloStatus[philID].left ==
1 && PhiloStatus[philID].right == 1) {        printf("Philosopher %d completed
his dinner\n", philID + 1);

       PhiloStatus[philID].left = PhiloStatus[philID].right = 10; // Mark as done

       int otherFork = philID - 1;
       if (otherFork == -1) otherFork = (n - 1);

       ForkAvail[philID].taken = ForkAvail[otherFork].taken = 0; // Release forks
printf("Philosopher %d released fork %d and fork %d\n", philID + 1, philID + 1, otherFork + 1);
completedPhilo++;
   }
   // Case: Left fork is taken, try for right     else if (PhiloStatus[philID].left == 1
&& PhiloStatus[philID].right == 0) {        if (philID == (n - 1)) {
           if (ForkAvail[philID].taken == 0) {
               ForkAvail[philID].taken = PhiloStatus[philID].right = 1;
               printf("Fork %d taken by Philosopher %d\n", philID + 1, philID + 1);
           } else {               printf("Philosopher %d is waiting for fork %d\n", philID + 1,
philID + 1);
           }
```

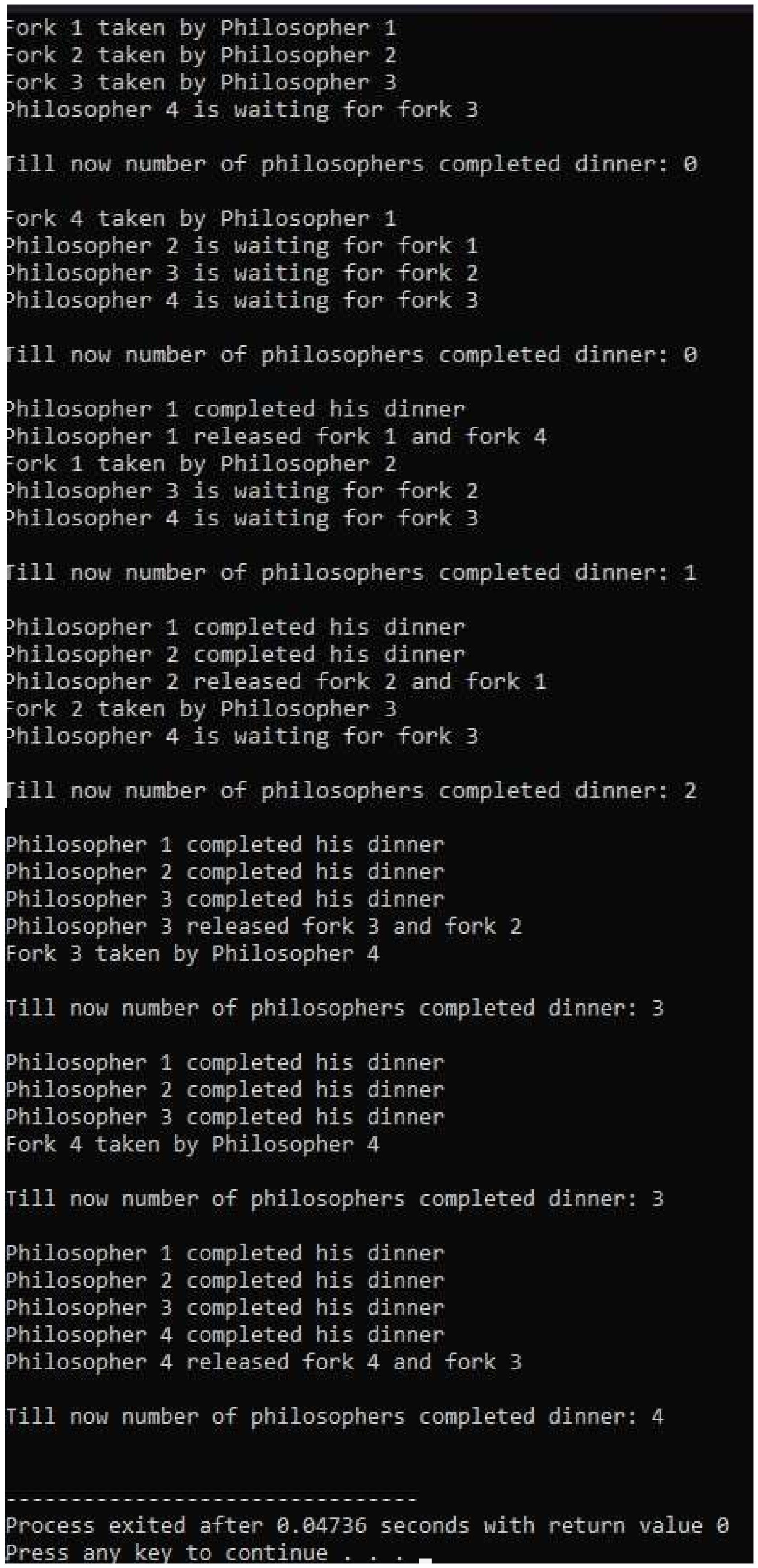
```
        } else {            int dupPhilID = philID;
philID -= 1;            if (philID == -1) philID =
(n - 1);

            if (ForkAvail[philID].taken == 0) {
                ForkAvail[philID].taken = PhiloStatus[dupPhilID].right = 1;
                printf("Fork %d taken by Philosopher %d\n", philID + 1, dupPhilID + 1);
            } else {                printf("Philosopher %d is waiting for fork %d\n", dupPhilID + 1,
philID + 1);
            }
        }
    }
    // Case: No forks taken yet     else if
(PhiloStatus[philID].left == 0) {       if (philID
== (n - 1)) {
            if (ForkAvail[philID - 1].taken == 0) {
                ForkAvail[philID - 1].taken = PhiloStatus[philID].left = 1;
printf("Fork %d taken by Philosopher %d\n", philID, philID + 1);
            } else {                printf("Philosopher %d is waiting for fork %d\n", philID +
1, philID);
            }
        } else {
            if (ForkAvail[philID].taken == 0) {
                ForkAvail[philID].taken = PhiloStatus[philID].left = 1;
printf("Fork %d taken by Philosopher %d\n", philID + 1, philID + 1);
            } else {                printf("Philosopher %d is waiting for fork %d\n", philID + 1,
philID + 1);
            }
        }
    }
}
int main() {     for (i = 0; i < n;
i++) {        ForkAvail[i].taken
= 0;
        PhiloStatus[i].left = 0;
        PhiloStatus[i].right = 0;
    }

    while (completedPhilo < n) {        for (i
= 0; i < n; i++) {            goForDinner(i);
        }
        printf("\nTill now number of philosophers completed dinner: %d\n\n", completedPhilo);
    }

    return 0; }
```

OUTPUT:



```
ork 1 taken by Philosopher 1
ork 2 taken by Philosopher 2
ork 3 taken by Philosopher 3
hilosopher 4 is waiting for fork 3

Till now number of philosophers completed dinner: 0

ork 4 taken by Philosopher 1
hilosopher 2 is waiting for fork 1
hilosopher 3 is waiting for fork 2
hilosopher 4 is waiting for fork 3

Till now number of philosophers completed dinner: 0

hilosopher 1 completed his dinner
hilosopher 1 released fork 1 and fork 4
ork 1 taken by Philosopher 2
hilosopher 3 is waiting for fork 2
hilosopher 4 is waiting for fork 3

Till now number of philosophers completed dinner: 1

hilosopher 1 completed his dinner
hilosopher 2 completed his dinner
hilosopher 2 released fork 2 and fork 1
ork 2 taken by Philosopher 3
hilosopher 4 is waiting for fork 3

Till now number of philosophers completed dinner: 2

Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 3 completed his dinner
Philosopher 3 released fork 3 and fork 2
Fork 3 taken by Philosopher 4

Till now number of philosophers completed dinner: 3

Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 3 completed his dinner
Fork 4 taken by Philosopher 4

Till now number of philosophers completed dinner: 3

Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 3 completed his dinner
Philosopher 4 completed his dinner
Philosopher 4 released fork 4 and fork 3

Till now number of philosophers completed dinner: 4



--------------------------------
Process exited after 0.04736 seconds with return value 0
Press any key to continue . . .
```