Mohd. Aftab Alam
02013302717

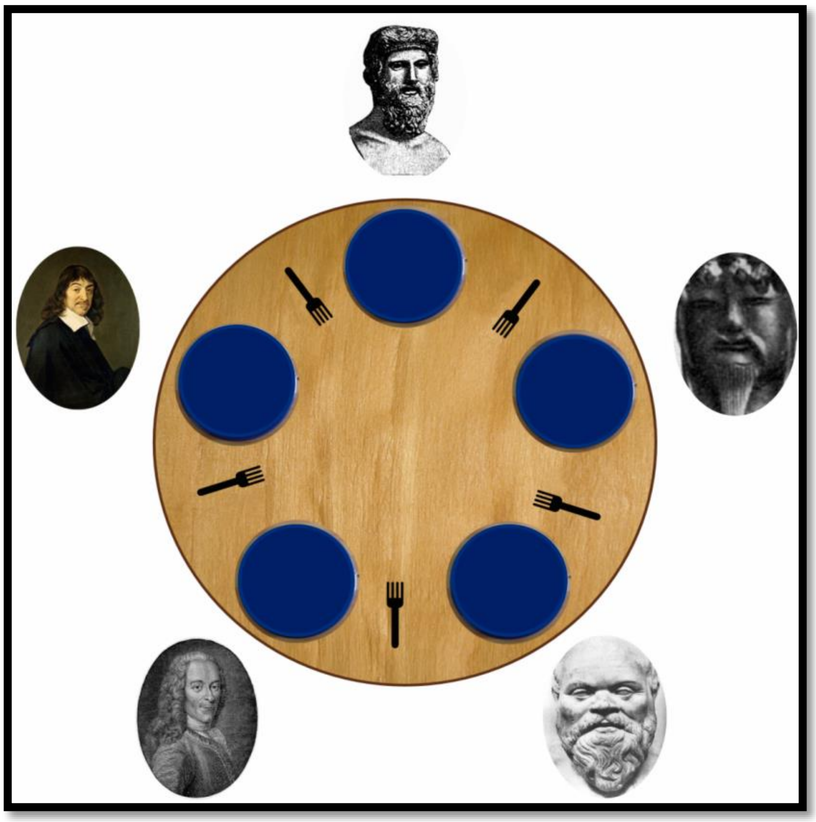
Dining Philosophers Problem

Aim :

Program To Implement Dining Philosophers Problem In C Programming Language.

Theory :

In Computer Science, the Dining Philosophers Problem is an example problem often used in concurrent algorithm design to illustrate synchronization issues and techniques for resolving them. It was originally formulated in **1965** by **Edsger Dijkstra** as a student exam exercise, presented in terms of computers competing for access to tape drive peripherals. Soon after, **Tony Hoare** gave the problem its present formulation.



Problem Statement

Five silent philosophers sit at a round table with bowls of spaghetti. Forks are placed between each pair of adjacent philosophers. Each philosopher must alternately think and eat. However, a philosopher can only eat spaghetti when they have both left and right forks. Each fork can be held by only one philosopher and so a philosopher can use the fork only if it is not being used by another philosopher. After an individual philosopher finishes eating, they need to put down both forks so that the forks become available to others. A philosopher can only take the fork on their right or the one on their left as they become available and they cannot start eating before getting both forks. Eating is not limited by the remaining amounts of spaghetti or stomach space; an infinite supply and an infinite demand are assumed. The problem is how to design a discipline of behaviour (algorithm) such that no philosopher will starve; i.e., each can forever continue to alternate between eating and thinking, assuming that no philosopher can know when others may want to eat or think.

Mohd. Aftab Alam
02013302717

Problems

The problem was designed to illustrate the challenges of avoiding **Deadlock,** a system state in which no progress is possible. To see that a proper solution to this problem is not obvious, consider a proposal in which each philosopher is instructed to behave as follows :

- think until the left fork is available; when it is, pick it up;
- think until the right fork is available; when it is, pick it up;
- when both forks are held, eat for a fixed amount of time;
- then, put the right fork down;
- then, put the left fork down;
- repeat from the beginning.

This attempted solution fails because it allows the system to reach a deadlock state, in which no progress is possible. This is a state in which each philosopher has picked up the fork to the left, and is waiting for the fork to the right to become available. With the given instructions, this state can be reached, and when it is reached, each philosopher will eternally wait for another (to the right) to release a fork.

Resource Starvation might also occur independently of deadlock if a particular philosopher is unable to acquire both forks because of a timing problem. For example, there might be a rule that the philosophers put down a fork after waiting ten minutes for the other fork to become available and wait a further ten minutes before making their next attempt. This scheme eliminates the possibility of deadlock (the system can always advance to a different state) but still suffers from the problem of live lock. If all five philosophers appear in the dining room at exactly the same time and each picks up the left fork at the same time the philosophers will wait ten minutes until they all put their forks down and then wait a further ten minutes before they all pick them up again.

Mutual Exclusion is the basic idea of the problem; the dining philosophers create a generic and abstract scenario useful for explaining issues of this type. The failures these philosophers may experience are analogous to the difficulties that arise in real computer programming when multiple programs need exclusive access to shared resources. These issues are studied in concurrent programming. The original problems of **Dijkstra** were related to external devices like tape drives. However, the difficulties exemplified by the Dining Philosophers Problem arise far more often when multiple processes access sets of data that are being updated. Complex systems such as operating system kernels use thousands of locks and synchronizations that require strict adherence to methods and protocols if such problems as deadlock, starvation, and data corruption are to be avoided.

Solutions

Solutions for Dining Philosophers Problem are listed below :

- **Resource Hierarchy Solution**
- **Arbitrator Solution**
- **Chandy / Misra Solution**

Mohd. Aftab Alam
02013302717

Code :

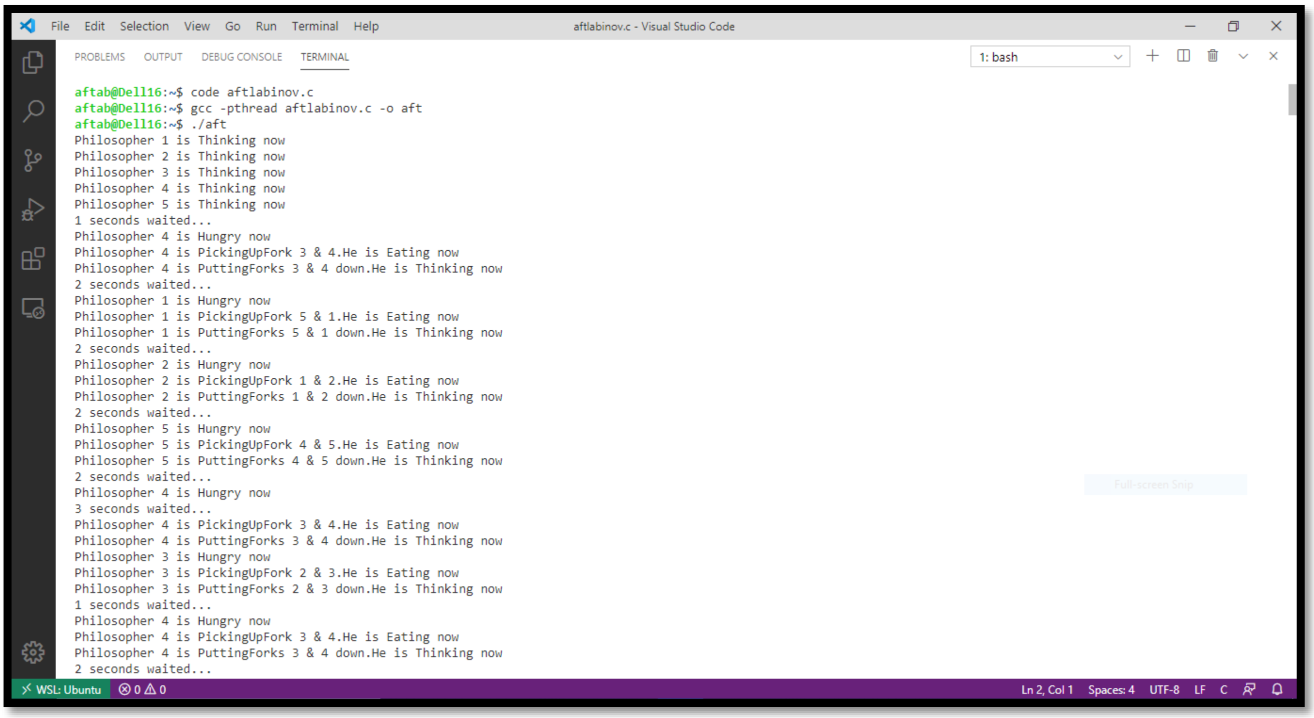
```c
#include <pthread.h>
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#define PHILOSOPERS_NUM 5
//philosophers are THINKING, then become HUNGRY so they start EATING
#define THINKING 0
#define HUNGRY 1
#define EATING 2
#define LEFT_PHILOSOPHER (philosopher_number + 4) % PHILOSOPERS_NUM
#define RIGHT_PHILOSOPHER (philosopher_number + 1) % PHILOSOPERS_NUM
pthread_mutex_t mutex;
pthread_cond_t cond_var[PHILOSOPERS_NUM];
pthread_t threadID[PHILOSOPERS_NUM];
//declare the functions
void * philospher(void *num);
void pickup_forks(int);
void return_forks(int);
void philosopher_can_eat(int);
int state[PHILOSOPERS_NUM];
int philosophers[PHILOSOPERS_NUM] = { 0, 1, 2, 3, 4 };
void *philospher(void *num) {
    while(1) {
        int x = rand() % 3;
        int *i = num;
        sleep(x + 1);
        printf("%d seconds waited...\n", x + 1);
        pickup_forks(*i);
        return_forks(*i); }
}
void pickup_forks(int philosopher_number) {
    //lock first
    pthread_mutex_lock (&mutex);
    //make philosopher HUNGRY
    state[philosopher_number] = HUNGRY;
    printf("Philosopher %d is Hungry now \n", philosopher_number + 1);
    //get what the philosopher is doing
    philosopher_can_eat(philosopher_number);
    //wait if he is EATING
    if (state[philosopher_number] != EATING) {
    pthread_cond_wait(&cond_var[philosopher_number], &mutex); }
    //else, unlock and sleep for 5 seconds
    pthread_mutex_unlock (&mutex);
}
```

Mohd. Aftab Alam
02013302717

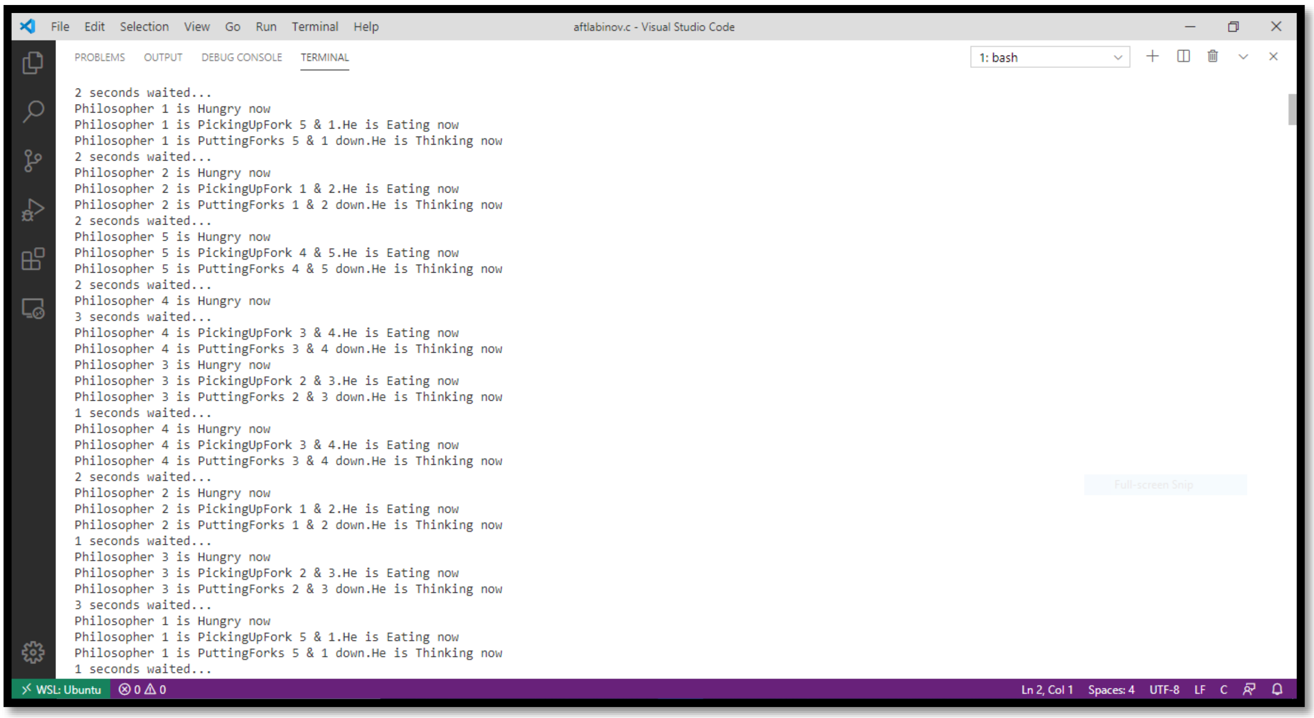
```
void return_forks(int philosopher_number)
{
    //lock first
    pthread_mutex_lock (&mutex);
    //make philosopher THINKING
    state[philosopher_number] = THINKING;
    printf("Philosopher %d is PuttingForks %d & %d down.He is Thinking
    now \n",philosopher_number+1,LEFT_PHILOSOPHER+1,philosopher_number+1);
    philosopher_can_eat(LEFT_PHILOSOPHER);
    philosopher_can_eat(RIGHT_PHILOSOPHER);
    pthread_mutex_unlock (&mutex);
}
void philosopher_can_eat(int philosopher_number)
{
    //check the state of the philosophers
    //if a philosopher is HUNGRY and his LEFT_PHILOSOPHER
    and RIGHT_PHILOSOPHER are not EATING then he can eat
    if (state[philosopher_number] == HUNGRY && state[LEFT_PHILOSOPHER]
    != EATING && state[RIGHT_PHILOSOPHER] != EATING) {
    state[philosopher_number] = EATING;
    printf("Philosopher %d is PickingUpFork %d & %d.He is Eating now \n",
    philosopher_number+1,LEFT_PHILOSOPHER+1,philosopher_number+1);
    pthread_cond_signal(&cond_var[philosopher_number]); }
}

int main()
{
    int i;
    time_t t;
    srand((unsigned) time(&t));
    pthread_t threadID[PHILOSOPERS_NUM];
    //set them all to THINKING initially
    for( i = 0; i < PHILOSOPERS_NUM; i++) {
        state[i] = THINKING;
        pthread_cond_init(&cond_var[i], NULL);
    }
    pthread_mutex_init (&mutex, NULL);
    for( i = 0; i < PHILOSOPERS_NUM; i++) {
        //create a pthread
        pthread_create(&threadID[i], NULL, philospher, &philosophers[i]);
        printf("Philosopher %d is Thinking now \n", i+1);
    }
    for( i = 0; i < PHILOSOPERS_NUM; i++) {
        pthread_join(threadID[i], NULL);
    }
    pthread_mutex_destroy(&mutex);
}
```

Mohd. Aftab Alam
02013302717

Screen Shots :


File Edit Selection View Go Run Terminal Help
aftlabinov.c - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: bash
WSL: Ubuntu
Ln 2, Col 1 Spaces: 4 UTF-8 LF C

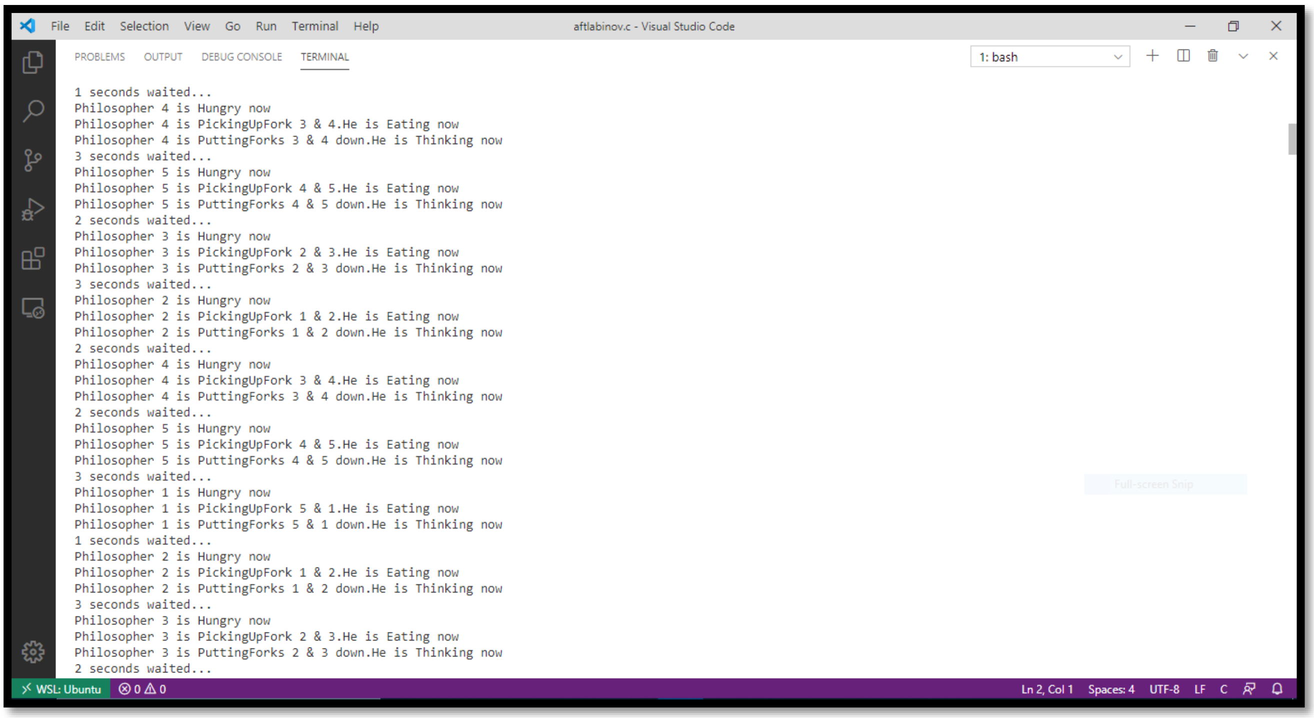
```
aftab@Dell16:~$ code aftlabinov.c
aftab@Dell16:~$ gcc -pthread aftlabinov.c -o aft
aftab@Dell16:~$ ./aft
Philosopher 1 is Thinking now
Philosopher 2 is Thinking now
Philosopher 3 is Thinking now
Philosopher 4 is Thinking now
Philosopher 5 is Thinking now
1 seconds waited...
Philosopher 4 is Hungry now
Philosopher 4 is PickingUpFork 3 & 4.He is Eating now
Philosopher 4 is PuttingForks 3 & 4 down.He is Thinking now
2 seconds waited...
Philosopher 1 is Hungry now
Philosopher 1 is PickingUpFork 5 & 1.He is Eating now
Philosopher 1 is PuttingForks 5 & 1 down.He is Thinking now
2 seconds waited...
Philosopher 2 is Hungry now
Philosopher 2 is PickingUpFork 1 & 2.He is Eating now
Philosopher 2 is PuttingForks 1 & 2 down.He is Thinking now
2 seconds waited...
Philosopher 5 is Hungry now
Philosopher 5 is PickingUpFork 4 & 5.He is Eating now
Philosopher 5 is PuttingForks 4 & 5 down.He is Thinking now
2 seconds waited...
Philosopher 4 is Hungry now
3 seconds waited...
Philosopher 4 is PickingUpFork 3 & 4.He is Eating now
Philosopher 4 is PuttingForks 3 & 4 down.He is Thinking now
Philosopher 3 is Hungry now
Philosopher 3 is PickingUpFork 2 & 3.He is Eating now
Philosopher 3 is PuttingForks 2 & 3 down.He is Thinking now
1 seconds waited...
Philosopher 4 is Hungry now
Philosopher 4 is PickingUpFork 3 & 4.He is Eating now
Philosopher 4 is PuttingForks 3 & 4 down.He is Thinking now
2 seconds waited...
```


File Edit Selection View Go Run Terminal Help
aftlabinov.c - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: bash
WSL: Ubuntu
Ln 2, Col 1 Spaces: 4 UTF-8 LF C

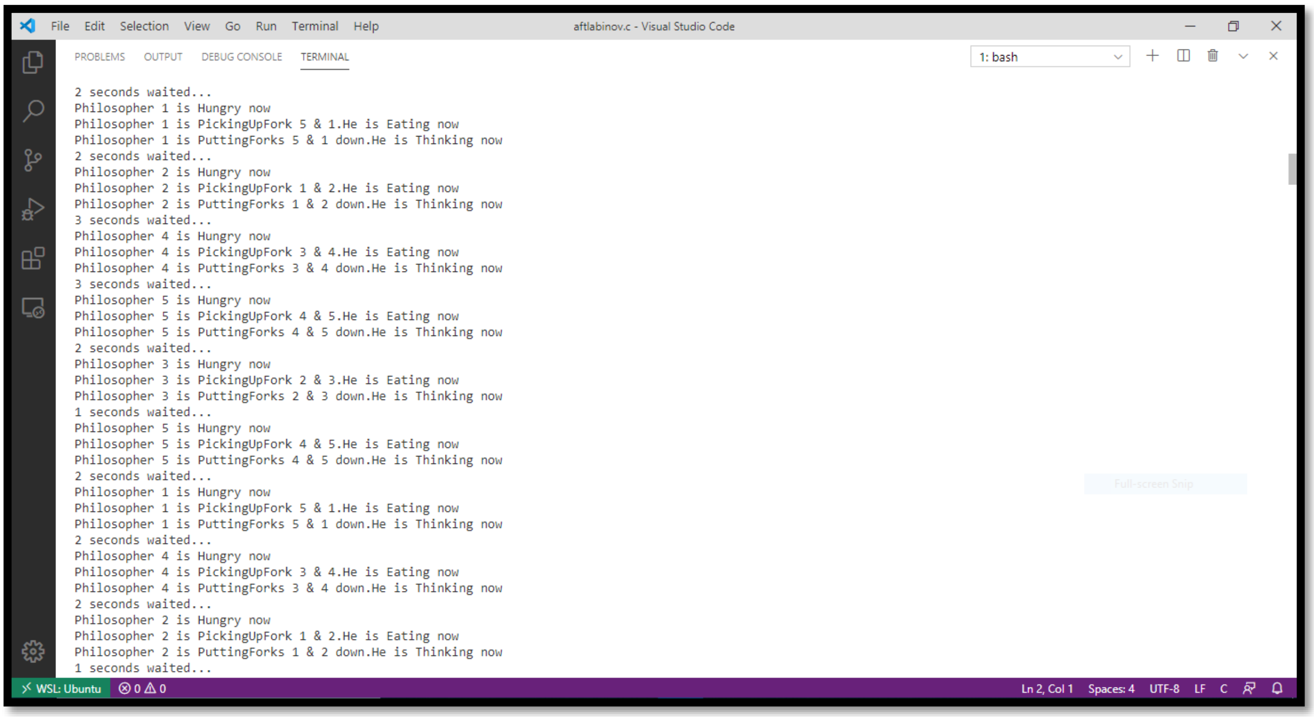
```
2 seconds waited...
Philosopher 1 is Hungry now
Philosopher 1 is PickingUpFork 5 & 1.He is Eating now
Philosopher 1 is PuttingForks 5 & 1 down.He is Thinking now
2 seconds waited...
Philosopher 2 is Hungry now
Philosopher 2 is PickingUpFork 1 & 2.He is Eating now
Philosopher 2 is PuttingForks 1 & 2 down.He is Thinking now
2 seconds waited...
Philosopher 5 is Hungry now
Philosopher 5 is PickingUpFork 4 & 5.He is Eating now
Philosopher 5 is PuttingForks 4 & 5 down.He is Thinking now
2 seconds waited...
Philosopher 4 is Hungry now
3 seconds waited...
Philosopher 4 is PickingUpFork 3 & 4.He is Eating now
Philosopher 4 is PuttingForks 3 & 4 down.He is Thinking now
Philosopher 3 is Hungry now
Philosopher 3 is PickingUpFork 2 & 3.He is Eating now
Philosopher 3 is PuttingForks 2 & 3 down.He is Thinking now
1 seconds waited...
Philosopher 4 is Hungry now
Philosopher 4 is PickingUpFork 3 & 4.He is Eating now
Philosopher 4 is PuttingForks 3 & 4 down.He is Thinking now
2 seconds waited...
Philosopher 2 is Hungry now
Philosopher 2 is PickingUpFork 1 & 2.He is Eating now
Philosopher 2 is PuttingForks 1 & 2 down.He is Thinking now
1 seconds waited...
Philosopher 3 is Hungry now
Philosopher 3 is PickingUpFork 2 & 3.He is Eating now
Philosopher 3 is PuttingForks 2 & 3 down.He is Thinking now
3 seconds waited...
Philosopher 1 is Hungry now
Philosopher 1 is PickingUpFork 5 & 1.He is Eating now
Philosopher 1 is PuttingForks 5 & 1 down.He is Thinking now
1 seconds waited...
```

Mohd. Aftab Alam
02013302717

Screen Shots :


File Edit Selection View Go Run Terminal Help
aftlabinov.c - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: bash
WSL: Ubuntu
Ln 2, Col 1 Spaces: 4 UTF-8 LF C

```
1 seconds waited...
Philosopher 4 is Hungry now
Philosopher 4 is PickingUpFork 3 & 4.He is Eating now
Philosopher 4 is PuttingForks 3 & 4 down.He is Thinking now
3 seconds waited...
Philosopher 5 is Hungry now
Philosopher 5 is PickingUpFork 4 & 5.He is Eating now
Philosopher 5 is PuttingForks 4 & 5 down.He is Thinking now
2 seconds waited...
Philosopher 3 is Hungry now
Philosopher 3 is PickingUpFork 2 & 3.He is Eating now
Philosopher 3 is PuttingForks 2 & 3 down.He is Thinking now
3 seconds waited...
Philosopher 2 is Hungry now
Philosopher 2 is PickingUpFork 1 & 2.He is Eating now
Philosopher 2 is PuttingForks 1 & 2 down.He is Thinking now
2 seconds waited...
Philosopher 4 is Hungry now
Philosopher 4 is PickingUpFork 3 & 4.He is Eating now
Philosopher 4 is PuttingForks 3 & 4 down.He is Thinking now
2 seconds waited...
Philosopher 5 is Hungry now
Philosopher 5 is PickingUpFork 4 & 5.He is Eating now
Philosopher 5 is PuttingForks 4 & 5 down.He is Thinking now
3 seconds waited...
Philosopher 1 is Hungry now
Philosopher 1 is PickingUpFork 5 & 1.He is Eating now
Philosopher 1 is PuttingForks 5 & 1 down.He is Thinking now
1 seconds waited...
Philosopher 2 is Hungry now
Philosopher 2 is PickingUpFork 1 & 2.He is Eating now
Philosopher 2 is PuttingForks 1 & 2 down.He is Thinking now
3 seconds waited...
Philosopher 3 is Hungry now
Philosopher 3 is PickingUpFork 2 & 3.He is Eating now
Philosopher 3 is PuttingForks 2 & 3 down.He is Thinking now
2 seconds waited...
```


File Edit Selection View Go Run Terminal Help
aftlabinov.c - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: bash
WSL: Ubuntu
Ln 2, Col 1 Spaces: 4 UTF-8 LF C

```
2 seconds waited...
Philosopher 1 is Hungry now
Philosopher 1 is PickingUpFork 5 & 1.He is Eating now
Philosopher 1 is PuttingForks 5 & 1 down.He is Thinking now
2 seconds waited...
Philosopher 2 is Hungry now
Philosopher 2 is PickingUpFork 1 & 2.He is Eating now
Philosopher 2 is PuttingForks 1 & 2 down.He is Thinking now
3 seconds waited...
Philosopher 4 is Hungry now
Philosopher 4 is PickingUpFork 3 & 4.He is Eating now
Philosopher 4 is PuttingForks 3 & 4 down.He is Thinking now
3 seconds waited...
Philosopher 5 is Hungry now
Philosopher 5 is PickingUpFork 4 & 5.He is Eating now
Philosopher 5 is PuttingForks 4 & 5 down.He is Thinking now
2 seconds waited...
Philosopher 3 is Hungry now
Philosopher 3 is PickingUpFork 2 & 3.He is Eating now
Philosopher 3 is PuttingForks 2 & 3 down.He is Thinking now
1 seconds waited...
Philosopher 5 is Hungry now
Philosopher 5 is PickingUpFork 4 & 5.He is Eating now
Philosopher 5 is PuttingForks 4 & 5 down.He is Thinking now
2 seconds waited...
Philosopher 1 is Hungry now
Philosopher 1 is PickingUpFork 5 & 1.He is Eating now
Philosopher 1 is PuttingForks 5 & 1 down.He is Thinking now
2 seconds waited...
Philosopher 4 is Hungry now
Philosopher 4 is PickingUpFork 3 & 4.He is Eating now
Philosopher 4 is PuttingForks 3 & 4 down.He is Thinking now
2 seconds waited...
Philosopher 2 is Hungry now
Philosopher 2 is PickingUpFork 1 & 2.He is Eating now
Philosopher 2 is PuttingForks 1 & 2 down.He is Thinking now
1 seconds waited...
```

Result : Program For Dining Philosophers Problem Was Implemented Successfully.