**Experiment1**

91981@Ananya ~

$ mkdir -p dir2

91981@Ananya ~

$

if [ -d "dir1" ]; then

echo "Directory exists."

else

echo "Directory does not exist."

fi

Directory exists.

91981@Ananya ~

$ pwd dict.txt

/home/91981

91981@Ananya ~

$ cd

91981@Ananya ~

$ ls -l dict2

ls: cannot access 'dict2': No such file or directory

91981@Ananya ~

$ ls -l dir2

total 0

91981@Ananya ~

$ ls -l dir1

total 0

-rw-r--r-- 1 91981 91981 0 Aug 5 13:42 dict.txt

91981@Ananya ~

$ ls -d

.

91981@Ananya ~

$ ls -a

. .. .bash\_history .bash\_profile .bashrc .inputrc .profile BoxDemo.txt Dict.txt dir1 dir2

91981@Ananya ~

$ ls -d

.

91981@Ananya ~

$ ls -d dict.txt

dict.txt

91981@Ananya ~

$ ls -a dict2

ls: cannot access 'dict2': No such file or directory

91981@Ananya ~

$ ls -a

. .. .bash\_history .bash\_profile .bashrc .inputrc .profile BoxDemo.txt Dict.txt dir1 dir2

91981@Ananya ~

$ rm -f

91981@Ananya ~

$ rm -i

rm: missing operand

Try 'rm --help' for more information.

91981@Ananya ~

$ rm --help

Usage: rm [OPTION]... [FILE]...

Remove (unlink) the FILE(s).

-f, --force ignore nonexistent files and arguments, never prompt

-i prompt before every removal

-I prompt once before removing more than three files, or

when removing recursively; less intrusive than -i,

while still giving protection against most mistakes

--interactive[=WHEN] prompt according to WHEN: never, once (-I), or

always (-i); without WHEN, prompt always

--one-file-system when removing a hierarchy recursively, skip any

directory that is on a file system different from

that of the corresponding command line argument

--no-preserve-root do not treat '/' specially

--preserve-root[=all] do not remove '/' (default);

with 'all', reject any command line argument

on a separate device from its parent

-r, -R, --recursive remove directories and their contents recursively

-d, --dir remove empty directories

-v, --verbose explain what is being done

--help display this help and exit

--version output version information and exit

By default, rm does not remove directories. Use the --recursive (-r or -R)

option to remove each listed directory, too, along with all of its contents.

To remove a file whose name starts with a '-', for example '-foo',

use one of these commands:

rm -- -foo

rm ./-foo

Note that if you use rm to remove a file, it might be possible to recover

some of its contents, given sufficient expertise and/or time. For greater

assurance that the contents are truly unrecoverable, consider using shred(1).

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>

Report any translation bugs to <https://translationproject.org/team/>

Full documentation <https://www.gnu.org/software/coreutils/rm>

or available locally via: info '(coreutils) rm invocation'

91981@Ananya ~

$ ls -p

BoxDemo.txt/ Dict.txt/ dir1/ dir2/

91981@Ananya ~

$ ls -u

dir1 dir2 Dict.txt BoxDemo.txt

91981@Ananya ~

$ ls -i

12666373952150031 BoxDemo.txt 9288674231635721 Dict.txt 9288674231635715 dir1 9851624185057028 dir2

91981@Ananya ~

$ rm -f

91981@Ananya ~

$ rm -i

rm: missing operand

Try 'rm --help' for more information.

91981@Ananya ~

$ rm --help

Usage: rm [OPTION]... [FILE]...

Remove (unlink) the FILE(s).

-f, --force ignore nonexistent files and arguments, never prompt

-i prompt before every removal

-I prompt once before removing more than three files, or

when removing recursively; less intrusive than -i,

while still giving protection against most mistakes

--interactive[=WHEN] prompt according to WHEN: never, once (-I), or

always (-i); without WHEN, prompt always

--one-file-system when removing a hierarchy recursively, skip any

directory that is on a file system different from

that of the corresponding command line argument

--no-preserve-root do not treat '/' specially

--preserve-root[=all] do not remove '/' (default);

with 'all', reject any command line argument

on a separate device from its parent

-r, -R, --recursive remove directories and their contents recursively

-d, --dir remove empty directories

-v, --verbose explain what is being done

--help display this help and exit

--version output version information and exit

By default, rm does not remove directories. Use the --recursive (-r or -R)

option to remove each listed directory, too, along with all of its contents.

To remove a file whose name starts with a '-', for example '-foo',

use one of these commands:

rm -- -foo

rm ./-foo

Note that if you use rm to remove a file, it might be possible to recover

some of its contents, given sufficient expertise and/or time. For greater

assurance that the contents are truly unrecoverable, consider using shred(1).

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>

Report any translation bugs to <https://translationproject.org/team/>

Full documentation <https://www.gnu.org/software/coreutils/rm>

or available locally via: info '(coreutils) rm invocation'

91981@Ananya ~

$ mv -f dir1 dir2

91981@Ananya ~

$ mv -i dir1 dir2

mv: cannot stat 'dir1': No such file or directory

91981@Ananya ~

$ mv -i dict.txt BoxDemo.txt

91981@Ananya ~

$

**Experiment2**

A black background with green and white text

Description automatically generated

**#!/bin/bash**

**echo "Enter number"**

**read number**

**fact=1**

**# Fix: Ensure spaces around brackets and variable**

**while [ $number -gt 0 ]**

**do**

**fact=$((fact \* number))**

**number=$((number - 1))**

**done**

**echo $fact**

A screen shot of a computer

Description automatically generated

A black screen with white text

Description automatically generated

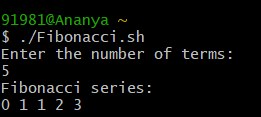
A black background with white text

Description automatically generated

**Experiment3**

A computer screen with white text

Description automatically generated



A computer screen shot of white text

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer screen shot of numbers and symbols

Description automatically generated

A black screen with white text

Description automatically generated

**Experiment4**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer screen shot of white text

Description automatically generated

A screenshot of a computer

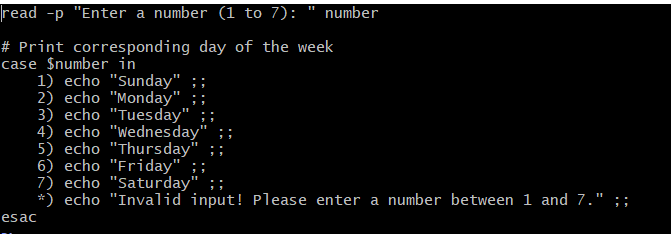
Description automatically generated

A computer screen shot of white text

Description automatically generated

A screenshot of a computer program

Description automatically generated



A screen shot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

**Experiment5**

**Code**

#include <iostream>

#include <iomanip> // For formatting output with precision

using namespace std;

// Function to calculate waiting time

void calculateWaitingTime(int processes[], int n, int burstTime[], int arrivalTime[], int waitingTime[]) {

int serviceTime[n];

serviceTime[0] = arrivalTime[0]; // Service time for first process

waitingTime[0] = 0; // Waiting time for first process is 0

// Calculate waiting time for each process

for (int i = 1; i < n; i++) {

// Service time is the time at which this process starts execution

serviceTime[i] = serviceTime[i - 1] + burstTime[i - 1];

// Waiting time = start time (service time) - arrival time

waitingTime[i] = serviceTime[i] - arrivalTime[i];

// If waiting time is negative, make it 0

if (waitingTime[i] < 0) {

waitingTime[i] = 0;

}

}

}

// Function to calculate turnaround time

void calculateTurnaroundTime(int processes[], int n, int burstTime[], int waitingTime[], int turnaroundTime[]) {

// Turnaround time = burst time + waiting time

for (int i = 0; i < n; i++) {

turnaroundTime[i] = burstTime[i] + waitingTime[i];

}

}

// Function to calculate average waiting time and turnaround time

void calculateAverageTime(int processes[], int n, int burstTime[], int arrivalTime[]) {

int waitingTime[n], turnaroundTime[n];

float totalWaitingTime = 0, totalTurnaroundTime = 0;

// Calculate waiting time for all processes

calculateWaitingTime(processes, n, burstTime, arrivalTime, waitingTime);

// Calculate turnaround time for all processes

calculateTurnaroundTime(processes, n, burstTime, waitingTime, turnaroundTime);

// Display the results

cout << "Process\tArrival Time\tBurst Time\tWaiting Time\tTurnaround Time\n";

for (int i = 0; i < n; i++) {

totalWaitingTime += waitingTime[i];

totalTurnaroundTime += turnaroundTime[i];

cout << "P" << i << "\t\t" << arrivalTime[i] << "\t\t" << burstTime[i]

<< "\t\t" << waitingTime[i] << "\t\t" << turnaroundTime[i] << "\n";

}

// Display average waiting and turnaround times

cout << fixed << setprecision(2); // Set precision to 2 decimal places

cout << "\nAverage Waiting Time: " << totalWaitingTime / n;

cout << "\nAverage Turnaround Time: " << totalTurnaroundTime / n << endl;

}

int main() {

// Number of processes

int n = 4;

// Process IDs (P0, P1, P2, P3)

int processes[] = {0, 1, 2, 3};

// Arrival times

int arrivalTime[] = {0, 2, 1, 3};

// Burst times

int burstTime[] = {5, 3, 8, 6};

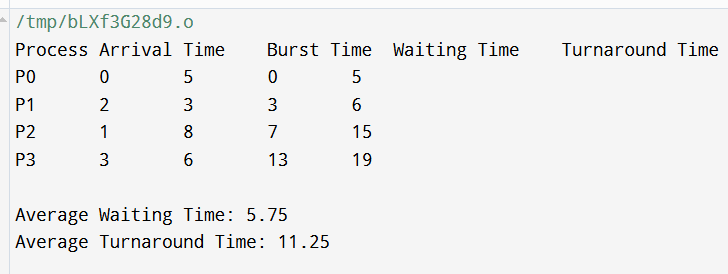
// Calculate and display the average time

calculateAverageTime(processes, n, burstTime, arrivalTime);

return 0;

}

Output



Code2

#include <stdio.h>

int main() {

// Number of processes

int n = 4;

// Process IDs

int process[] = {1, 2, 3, 4};

// Burst times of processes

int burst\_time[] = {5, 3, 8, 6};

// Arrays to store Completion Time, Turnaround Time, and Waiting Time

int completion\_time[4], turnaround\_time[4], waiting\_time[4];

// Calculate Completion Time (CT)

completion\_time[0] = burst\_time[0]; // First process finishes after its burst time

for (int i = 1; i < n; i++) {

completion\_time[i] = completion\_time[i-1] + burst\_time[i];

}

// Calculate Turnaround Time (TAT) and Waiting Time (WT)

for (int i = 0; i < n; i++) {

turnaround\_time[i] = completion\_time[i]; // Since Arrival Time is 0, TAT = CT

waiting\_time[i] = turnaround\_time[i] - burst\_time[i]; // WT = TAT - BT

}

// Calculate Average Turnaround Time and Waiting Time

float total\_tat = 0, total\_wt = 0;

for (int i = 0; i < n; i++) {

total\_tat += turnaround\_time[i];

total\_wt += waiting\_time[i];

}

float avg\_tat = total\_tat / n;

float avg\_wt = total\_wt / n;

// Display the results

printf("Process\tBurst Time\tCompletion Time\tTurnaround Time\tWaiting Time\n");

for (int i = 0; i < n; i++) {

printf("P%d\t%d\t\t%d\t\t%d\t\t%d\n", process[i], burst\_time[i], completion\_time[i], turnaround\_time[i], waiting\_time[i]);

}

printf("Average Turnaround Time: %.2f\n", avg\_tat);

printf("Average Waiting Time: %.2f\n", avg\_wt);

return 0;

}

Output

A screenshot of a computer

Description automatically generated

**Experiment6:**

**Code1**

**#include <stdio.h>**

**typedef struct {**

**int pid; // Process ID**

**int arrival; // Arrival Time**

**int burst; // Burst Time**

**int waiting; // Waiting Time**

**int turnaround; // Turnaround Time**

**} Process;**

**void calculateSJF(Process processes[], int n) {**

**int completed = 0, currentTime = 0, i;**

**float totalWaitingTime = 0, totalTurnaroundTime = 0;**

**while (completed < n) {**

**// Find the process with the shortest burst time among the available processes**

**int idx = -1;**

**int minBurst = 9999;**

**for (i = 0; i < n; i++) {**

**if (processes[i].arrival <= currentTime && processes[i].burst < minBurst && processes[i].burst > 0) {**

**minBurst = processes[i].burst;**

**idx = i;**

**}**

**}**

**if (idx != -1) { // If a process is found**

**currentTime += processes[idx].burst;**

**processes[idx].waiting = currentTime - processes[idx].arrival - processes[idx].burst;**

**processes[idx].turnaround = currentTime - processes[idx].arrival;**

**totalWaitingTime += processes[idx].waiting;**

**totalTurnaroundTime += processes[idx].turnaround;**

**processes[idx].burst = 0; // Mark process as completed**

**completed++;**

**} else {**

**currentTime++; // If no process is available, increment time**

**}**

**}**

**// Display results**

**printf("Process\tArrival\tBurst\tWaiting\tTurnaround\n");**

**for (i = 0; i < n; i++) {**

**printf("P%d\t%d\t%d\t%d\t%d\n", processes[i].pid, processes[i].arrival, processes[i].burst, processes[i].waiting, processes[i].turnaround);**

**}**

**printf("Average Waiting Time: %.2f\n", totalWaitingTime / n);**

**printf("Average Turnaround Time: %.2f\n", totalTurnaroundTime / n);**

**}**

**int main() {**

**Process processes[] = {**

**{1, 2, 6, 0, 0},**

**{2, 5, 2, 0, 0},**

**{3, 1, 8, 0, 0},**

**{4, 0, 3, 0, 0},**

**{5, 4, 4, 0, 0}**

**};**

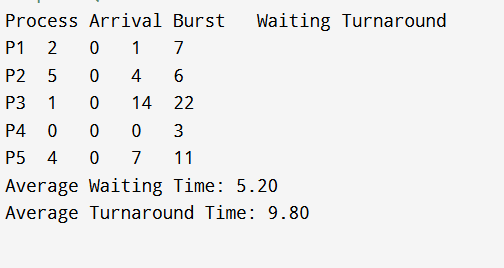
**int n = sizeof(processes) / sizeof(processes[0]);**

**calculateSJF(processes, n);**

**return 0;**

**}**

**Output**



**Code2**

**Srtf algorithm**

**#include <stdio.h>**

**typedef struct {**

**int pid; // Process ID**

**int arrival; // Arrival Time**

**int burst; // Burst Time**

**int remaining; // Remaining Time**

**int waiting; // Waiting Time**

**int turnaround; // Turnaround Time**

**} Process;**

**void calculateSJTF(Process processes[], int n) {**

**int completed = 0, currentTime = 0, i;**

**float totalWaitingTime = 0, totalTurnaroundTime = 0;**

**// Initialize remaining burst times**

**for (i = 0; i < n; i++) {**

**processes[i].remaining = processes[i].burst;**

**}**

**while (completed < n) {**

**// Find the process with the shortest remaining time among the available processes**

**int idx = -1;**

**int minRemaining = 9999;**

**for (i = 0; i < n; i++) {**

**if (processes[i].arrival <= currentTime && processes[i].remaining > 0 && processes[i].remaining < minRemaining) {**

**minRemaining = processes[i].remaining;**

**idx = i;**

**}**

**}**

**if (idx != -1) { // If a process is found**

**processes[idx].remaining--; // Process is executed for one time unit**

**currentTime++;**

**if (processes[idx].remaining == 0) { // Process completed**

**completed++;**

**processes[idx].turnaround = currentTime - processes[idx].arrival;**

**processes[idx].waiting = processes[idx].turnaround - processes[idx].burst;**

**totalWaitingTime += processes[idx].waiting;**

**totalTurnaroundTime += processes[idx].turnaround;**

**}**

**} else {**

**currentTime++; // If no process is available, increment time**

**}**

**}**

**// Display results**

**printf("Process\tArrival\tBurst\tWaiting\tTurnaround\n");**

**for (i = 0; i < n; i++) {**

**printf("P%d\t%d\t%d\t%d\t%d\n", processes[i].pid, processes[i].arrival, processes[i].burst, processes[i].waiting, processes[i].turnaround);**

**}**

**printf("Average Waiting Time: %.2f\n", totalWaitingTime / n);**

**printf("Average Turnaround Time: %.2f\n", totalTurnaroundTime / n);**

**}**

**int main() {**

**Process processes[] = {**

**{1, 2, 6, 0, 0, 0},**

**{2, 5, 2, 0, 0, 0},**

**{3, 1, 8, 0, 0, 0},**

**{4, 0, 3, 0, 0, 0},**

**{5, 4, 4, 0, 0, 0}**

**};**

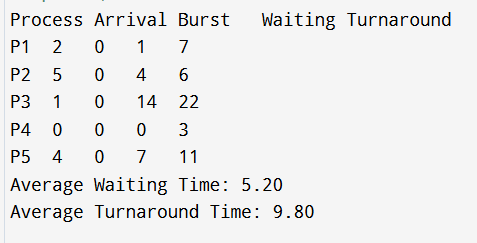
**int n = sizeof(processes) / sizeof(processes[0]);**

**calculateSJTF(processes, n);**

**return 0;**

**}**

Output



**Experiment7:**

**code**

#include <stdio.h>

struct Process {

char name[3];

int burst\_time;

int priority;

int waiting\_time;

int turnaround\_time;

};

// Function to sort processes based on priority (ascending order)

void sort\_by\_priority(struct Process proc[], int n) {

for (int i = 0; i < n - 1; i++) {

for (int j = i + 1; j < n; j++) {

if (proc[i].priority > proc[j].priority) {

// Swap the processes

struct Process temp = proc[i];

proc[i] = proc[j];

proc[j] = temp;

}

}

}

}

// Function to calculate waiting time and turnaround time

void calculate\_times(struct Process proc[], int n) {

int current\_time = 0;

for (int i = 0; i < n; i++) {

// Waiting time is the current time before the process starts

proc[i].waiting\_time = current\_time;

// Turnaround time is waiting time + burst time

proc[i].turnaround\_time = proc[i].waiting\_time + proc[i].burst\_time;

// Move current time forward by burst time of the current process

current\_time += proc[i].burst\_time;

}

}

// Function to display the processes and their calculated times

void display(struct Process proc[], int n) {

printf("Process\tPriority\tBurst Time\tWaiting Time\tTurnaround Time\n");

for (int i = 0; i < n; i++) {

printf("%s\t%d\t\t%d\t\t%d\t\t%d\n",

proc[i].name, proc[i].priority, proc[i].burst\_time,

proc[i].waiting\_time, proc[i].turnaround\_time);

}

}

int main() {

// Initialize the processes with their burst times and priorities

struct Process processes[] = {

{"p3", 5, 1, 0, 0},

{"p1", 7, 2, 0, 0},

{"p4", 7, 3, 0, 0},

{"p2", 10, 0, 0, 0}

};

int n = sizeof(processes) / sizeof(processes[0]);

// Sort processes by priority

sort\_by\_priority(processes, n);

// Calculate waiting time and turnaround time

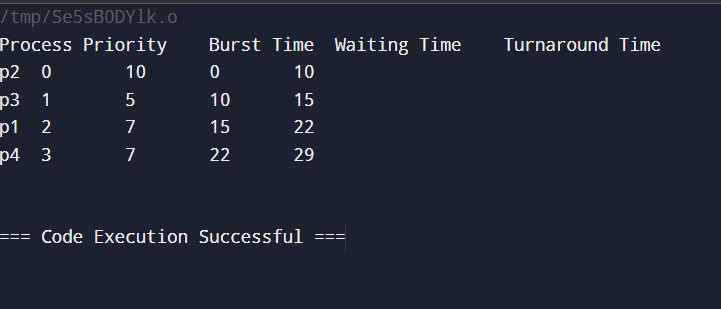
calculate\_times(processes, n);

// Display the results

display(processes, n);

return 0;

}



**Experiment8:**

#include <stdio.h>

#define MAX\_PROCESSES 4

// Structure to represent each process

struct Process {

int id; // Process ID

int bt; // Burst Time

int at; // Arrival Time

int remaining\_bt; // Remaining Burst Time

int wt; // Waiting Time

int tat; // Turnaround Time

int completion\_time; // Completion Time

};

void findWaitingTime(struct Process p[], int n, int quantum);

void findTurnaroundTime(struct Process p[], int n);

void findCompletionTime(struct Process p[], int n);

void findAverageTimes(struct Process p[], int n);

int main() {

// Processes with burst time and arrival time

struct Process p[MAX\_PROCESSES] = {

{1, 7, 0, 7, 0, 0, 0}, // p1: bt=7, at=0

{2, 10, 0, 10, 0, 0, 0}, // p2: bt=10, at=0

{3, 5, 0, 5, 0, 0, 0}, // p3: bt=5, at=0

{4, 3, 0, 3, 0, 0, 0} // p4: bt=3, at=0

};

int quantum = 4; // Time quantum

findWaitingTime(p, MAX\_PROCESSES, quantum);

findTurnaroundTime(p, MAX\_PROCESSES);

findCompletionTime(p, MAX\_PROCESSES);

findAverageTimes(p, MAX\_PROCESSES);

return 0;

}

// Function to find the waiting time of each process

void findWaitingTime(struct Process p[], int n, int quantum) {

int remaining\_processes = n;

int current\_time = 0;

int queue[MAX\_PROCESSES];

int front = 0, rear = 0;

// Enqueue all processes initially

for (int i = 0; i < n; i++) {

queue[rear++] = i;

}

while (remaining\_processes > 0) {

int i = queue[front++];

if (p[i].remaining\_bt > quantum) {

p[i].remaining\_bt -= quantum;

current\_time += quantum;

queue[rear++] = i;

} else {

current\_time += p[i].remaining\_bt;

p[i].wt = current\_time - p[i].at - p[i].bt;

p[i].remaining\_bt = 0;

remaining\_processes--;

}

}

}

// Function to find the turnaround time for each process

void findTurnaroundTime(struct Process p[], int n) {

for (int i = 0; i < n; i++) {

p[i].tat = p[i].bt + p[i].wt;

}

}

// Function to find the completion time for each process

void findCompletionTime(struct Process p[], int n) {

for (int i = 0; i < n; i++) {

p[i].completion\_time = p[i].at + p[i].wt + p[i].bt;

}

}

// Function to find average waiting time and turnaround time

void findAverageTimes(struct Process p[], int n) {

int total\_wt = 0, total\_tat = 0;

for (int i = 0; i < n; i++) {

total\_wt += p[i].wt;

total\_tat += p[i].tat;

}

printf("\nProcess ID\tBurst Time\tArrival Time\tWaiting Time\tTurnaround Time\tCompletion Time\n");

for (int i = 0; i < n; i++) {

printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n",

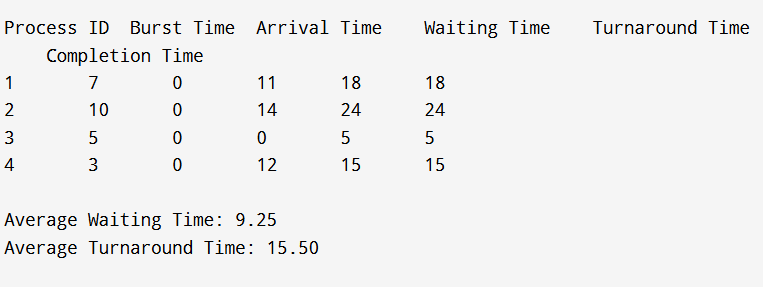
p[i].id, p[i].bt, p[i].at, p[i].wt, p[i].tat, p[i].completion\_time);

}

printf("\nAverage Waiting Time: %.2f", (float)total\_wt / n);

printf("\nAverage Turnaround Time: %.2f\n", (float)total\_tat / n);

}



**Experiment10:**

**Algorithm/Flowchart/Code/Sample Outputs**

**import threading**

**import time**

**import random**

**from threading import Semaphore**

**# Number of philosophers**

**N = 5**

**# Semaphores for forks (binary semaphores)**

**forks = [Semaphore(1) for \_ in range(N)]**

**# Semaphore to control the number of philosophers who can eat at the same time**

**# Ensures deadlock prevention (at most N-1 philosophers are allowed to pick up forks)**

**room = Semaphore(N - 1)**

**class Philosopher(threading.Thread):**

**def \_\_init\_\_(self, id):**

**threading.Thread.\_\_init\_\_(self)**

**self.id = id**

**def run(self):**

**while True:**

**# Philosopher is thinking**

**self.think()**

**# Philosopher tries to pick up forks**

**self.pick\_up\_forks()**

**# Philosopher is eating**

**self.eat()**

**# Philosopher puts down forks**

**self.put\_down\_forks()**

**def think(self):**

**print(f"Philosopher {self.id} is thinking.")**

**time.sleep(random.uniform(1, 3)) # Simulate thinking time**

**def pick\_up\_forks(self):**

**print(f"Philosopher {self.id} is hungry and trying to pick up forks.")**

**# Entering the room (at most N-1 philosophers allowed)**

**room.acquire()**

**# Pick up left fork**

**forks[self.id].acquire()**

**print(f"Philosopher {self.id} picked up left fork.")**

**# Pick up right fork**

**forks[(self.id + 1) % N].acquire()**

**print(f"Philosopher {self.id} picked up right fork.")**

**def eat(self):**

**print(f"Philosopher {self.id} is eating.")**

**time.sleep(random.uniform(1, 2)) # Simulate eating time**

**def put\_down\_forks(self):**

**# Put down right fork**

**forks[(self.id + 1) % N].release()**

**print(f"Philosopher {self.id} put down right fork.")**

**# Put down left fork**

**forks[self.id].release()**

**print(f"Philosopher {self.id} put down left fork.")**

**# Leave the room (allow another philosopher to enter)**

**room.release()**

**if \_\_name\_\_ == "\_\_main\_\_":**

**philosophers = [Philosopher(i) for i in range(N)]**

**for philosopher in philosophers:**

**philosopher.start()**

**for philosopher in philosophers:**

**philosopher.join()**

**output**

**Philosopher 0 is thinking.Philosopher 1 is thinking.**

**Philosopher 2 is thinking.**

**Philosopher 3 is thinking.**

**Philosopher 4 is thinking.**

**Philosopher 4 is hungry and trying to pick up forks.**

**Philosopher 4 picked up left fork.**

**Philosopher 4 picked up right fork.**

**Philosopher 4 is eating.**

**Philosopher 2 is hungry and trying to pick up forks.**

**Philosopher 2 picked up left fork.**

**Philosopher 2 picked up right fork.**

**Philosopher 2 is eating.**

**Philosopher 1 is hungry and trying to pick up forks.**

**Philosopher 1 picked up left fork.**

**Philosopher 0 is hungry and trying to pick up forks.**

**Philosopher 3 is hungry and trying to pick up forks.**

**Philosopher 4 put down right fork.Philosopher 0 picked up left fork.**

**Philosopher 4 put down left fork.**

**Philosopher 4 is thinking.**

**Philosopher 2 put down right fork.**

**Philosopher 2 put down left fork.**

**Philosopher 2 is thinking.**

**Philosopher 3 picked up left fork.**

**Philosopher 3 picked up right fork.**

**Philosopher 3 is eating.**

**Philosopher 1 picked up right fork.**

**Philosopher 1 is eating.**

**Philosopher 1 put down right fork.**

**Philosopher 1 put down left fork.**

**Philosopher 1 is thinking.**

**Philosopher 0 picked up right fork.**

**Philosopher 0 is eating.**

**Philosopher 3 put down right fork.**

**Philosopher 3 put down left fork.**

**Philosopher 3 is thinking.**

**Philosopher 2 is hungry and trying to pick up forks.**

**Philosopher 2 picked up left fork.**

**Philosopher 2 picked up right fork.**

**Philosopher 2 is eating.**

**Philosopher 4 is hungry and trying to pick up forks.**

**Philosopher 4 picked up left fork.**

**Philosopher 0 put down right fork.**

**Philosopher 0 put down left fork.**

**Philosopher 0 is thinking.**

**Philosopher 4 picked up right fork.**

**Philosopher 4 is eating.**

**Philosopher 1 is hungry and trying to pick up forks.**

**Philosopher 1 picked up left fork.**

**Philosopher 3 is hungry and trying to pick up forks.**

**Philosopher 2 put down right fork.Philosopher 3 picked up left fork.**

**Philosopher 2 put down left fork.**

**Philosopher 2 is thinking.**

**Philosopher 1 picked up right fork.**

**Philosopher 1 is eating.**

**Philosopher 4 put down right fork.**

**Philosopher 4 put down left fork.**

**Philosopher 4 is thinking.**

**Philosopher 3 picked up right fork.**

**Philosopher 3 is eating.**

**Philosopher 2 is hungry and trying to pick up forks.**

**Philosopher 0 is hungry and trying to pick up forks.**

**Philosopher 0 picked up left fork.**

**Philosopher 1 put down right fork.**

**Philosopher 1 put down left fork.**

**Philosopher 1 is thinking.**

**Philosopher 2 picked up left fork.**

**Philosopher 0 picked up right fork.**

**Philosopher 0 is eating.**

**Philosopher 4 is hungry and trying to pick up forks.**

**Philosopher 3 put down right fork.Philosopher 4 picked up left fork.**

**Philosopher 3 put down left fork.**

**Philosopher 3 is thinking.**

**Philosopher 2 picked up right fork.**

**Philosopher 2 is eating.**

**Philosopher 1 is hungry and trying to pick up forks.**

**Philosopher 0 put down right fork.Philosopher 1 picked up left fork.**

**Philosopher 0 put down left fork.**

**Philosopher 0 is thinking.**

**Philosopher 4 picked up right fork.**

**Philosopher 4 is eating.**

**Philosopher 2 put down right fork.**

**Philosopher 2 put down left fork.**

**Philosopher 2 is thinking.**

**Philosopher 1 picked up right fork.**

**Philosopher 1 is eating.**

**Philosopher 4 put down right fork.**

**Philosopher 4 put down left fork.**

**Philosopher 4 is thinking.**

**Philosopher 3 is hungry and trying to pick up forks.**

**Philosopher 3 picked up left fork.**

**Philosopher 3 picked up right fork.**

**Philosopher 3 is eating.**

**Philosopher 0 is hungry and trying to pick up forks.**

**Philosopher 0 picked up left fork.**

**Philosopher 1 put down right fork.**

**Philosopher 1 put down left fork.**

**Philosopher 1 is thinking.**

**Philosopher 0 picked up right fork.**

**Philosopher 0 is eating.**

**Philosopher 2 is hungry and trying to pick up forks.**

**Philosopher 2 picked up left fork.**

**Philosopher 3 put down right fork.**

**Philosopher 3 put down left fork.**

**Philosopher 3 is thinking.**

**Philosopher 2 picked up right fork.**

**Philosopher 2 is eating.**

**Philosopher 4 is hungry and trying to pick up forks.**

**Philosopher 4 picked up left fork.**

**Philosopher 3 is hungry and trying to pick up forks.**

**Philosopher 0 put down right fork.**

**Philosopher 0 put down left fork.**

**Philosopher 0 is thinking.**

**Philosopher 4 picked up right fork.**

**Philosopher 4 is eating.**

**Philosopher 2 put down right fork.**

**Philosopher 2 put down left fork.**

**Philosopher 2 is thinking.**

**Philosopher 3 picked up left fork.**

**Philosopher 1 is hungry and trying to pick up forks.**

**Philosopher 1 picked up left fork.**

**Philosopher 1 picked up right fork.**

**Philosopher 1 is eating.**

**Philosopher 4 put down right fork.**

**Philosopher 4 put down left fork.**

**Philosopher 4 is thinking.**

**Philosopher 3 picked up right fork.**

**Philosopher 3 is eating.**

**Philosopher 2 is hungry and trying to pick up forks.**

**Philosopher 0 is hungry and trying to pick up forks.**

**Philosopher 0 picked up left fork.**

**Philosopher 1 put down right fork.Philosopher 2 picked up left fork.**

**Philosopher 1 put down left fork.**

**Philosopher 1 is thinking.**

**Philosopher 0 picked up right fork.**

**Philosopher 0 is eating.**

**Philosopher 0 put down right fork.**

**Philosopher 0 put down left fork.**

**Philosopher 0 is thinking.**

**Philosopher 3 put down right fork.**

**Philosopher 3 put down left fork.**

**Philosopher 3 is thinking.**

**Philosopher 2 picked up right fork.**

**Philosopher 2 is eating.**

**Philosopher 4 is hungry and trying to pick up forks.**

**Philosopher 4 picked up left fork.**

**Philosopher 4 picked up right fork.**

**Philosopher 4 is eating.**

**Philosopher 3 is hungry and trying to pick up forks.**

**Philosopher 2 put down right fork.**

**Philosopher 2 put down left fork.**

**Philosopher 2 is thinking.**

**Philosopher 3 picked up left fork.**

**Philosopher 1 is hungry and trying to pick up forks.**

**Philosopher 1 picked up left fork.**

**Philosopher 1 picked up right fork.**

**Philosopher 1 is eating.**

**Philosopher 0 is hungry and trying to pick up forks.**

**Philosopher 4 put down right fork.**

**Philosopher 4 put down left fork.**

**Philosopher 4 is thinking.**

**Philosopher 0 picked up left fork.**

**Philosopher 3 picked up right fork.**

**Philosopher 3 is eating.**

**Philosopher 2 is hungry and trying to pick up forks.**

**Philosopher 4 is hungry and trying to pick up forks.**

**Philosopher 3 put down right fork.**

**Philosopher 3 put down left fork.**

**Philosopher 3 is thinking.**

**Philosopher 4 picked up left fork.**

**Philosopher 1 put down right fork.Philosopher 2 picked up left fork.**

**Philosopher 2 picked up right fork.**

**Philosopher 2 is eating.**

**Philosopher 1 put down left fork.**

**Philosopher 1 is thinking.**

**Philosopher 0 picked up right fork.**

**Philosopher 0 is eating.**

**Philosopher 0 put down right fork.**

**Philosopher 0 put down left fork.**

**Philosopher 0 is thinking.**

**Philosopher 4 picked up right fork.**

**Philosopher 4 is eating.**

**Philosopher 1 is hungry and trying to pick up forks.**

**Philosopher 1 picked up left fork.**

**Philosopher 2 put down right fork.**

**Philosopher 2 put down left fork.**

**Philosopher 2 is thinking.**

**Philosopher 1 picked up right fork.**

**Philosopher 1 is eating.**

**Philosopher 3 is hungry and trying to pick up forks.**

**Philosopher 3 picked up left fork.**

**Philosopher 0 is hungry and trying to pick up forks.**

**Philosopher 4 put down right fork.Philosopher 0 picked up left fork.**

**Philosopher 4 put down left fork.**

**Philosopher 4 is thinking.**

**Philosopher 3 picked up right fork.**

**Philosopher 3 is eating.**

**Philosopher 1 put down right fork.**

**Philosopher 1 put down left fork.**

**Philosopher 1 is thinking.**

**Philosopher 0 picked up right fork.**

**Philosopher 0 is eating.**

**Philosopher 2 is hungry and trying to pick up forks.**

**Philosopher 2 picked up left fork.**

**Philosopher 3 put down right fork.**

**Philosopher 3 put down left fork.**

**Philosopher 3 is thinking.**

**Philosopher 2 picked up right fork.**

**Philosopher 2 is eating.**

**Philosopher 0 put down right fork.**

**Philosopher 0 put down left fork.**

**Philosopher 0 is thinking.**

**Philosopher 4 is hungry and trying to pick up forks.**

**Philosopher 4 picked up left fork.**

**Philosopher 4 picked up right fork.**

**Philosopher 4 is eating.**

**Philosopher 1 is hungry and trying to pick up forks.**

**Philosopher 1 picked up left fork.**

**Philosopher 2 put down right fork.**

**Philosopher 2 put down left fork.**

**Philosopher 2 is thinking.**

**Philosopher 1 picked up right fork.**

**Philosopher 1 is eating.**

**Philosopher 4 put down right fork.**

**Philosopher 4 put down left fork.**

**Philosopher 4 is thinking.**

**Philosopher 0 is hungry and trying to pick up forks.**

**Philosopher 0 picked up left fork.**

**Philosopher 3 is hungry and trying to pick up forks.**

**Philosopher 3 picked up left fork.**

**Philosopher 3 picked up right fork.**

**Philosopher 3 is eating.**

**Philosopher 1 put down right fork.**

**Philosopher 1 put down left fork.**

**Philosopher 1 is thinking.**

**Philosopher 0 picked up right fork.**

**Philosopher 0 is eating.**

**Philosopher 4 is hungry and trying to pick up forks.**

**Philosopher 3 put down right fork.Philosopher 4 picked up left fork.**

**Philosopher 3 put down left fork.**

**Philosopher 3 is thinking.**

**Philosopher 1 is hungry and trying to pick up forks.**

**Philosopher 2 is hungry and trying to pick up forks.**

**Philosopher 2 picked up left fork.**

**Philosopher 2 picked up right fork.**

**Philosopher 2 is eating.**

**Philosopher 0 put down right fork.Philosopher 1 picked up left fork.**

**Philosopher 0 put down left fork.**

**Philosopher 0 is thinking.**

**Philosopher 4 picked up right fork.**

**Philosopher 4 is eating.**

**Philosopher 3 is hungry and trying to pick up forks.**

**Philosopher 2 put down right fork.Philosopher 3 picked up left fork.**

**Philosopher 2 put down left fork.**

**Philosopher 2 is thinking.**

**Philosopher 1 picked up right fork.**

**Philosopher 1 is eating.**

**Philosopher 4 put down right fork.**

**Philosopher 4 put down left fork.**

**Philosopher 4 is thinking.**

**Philosopher 3 picked up right fork.**

**Philosopher 3 is eating.**

**Philosopher 0 is hungry and trying to pick up forks.**

**Philosopher 0 picked up left fork.**

**Philosopher 1 put down right fork.**

**Philosopher 1 put down left fork.**

**Philosopher 1 is thinking.**

**Philosopher 0 picked up right fork.**

**Philosopher 0 is eating.**

**Philosopher 2 is hungry and trying to pick up forks.**

**Philosopher 2 picked up left fork.**

**Philosopher 3 put down right fork.**

**Philosopher 3 put down left fork.**

**Philosopher 3 is thinking.**

**Philosopher 2 picked up right fork.**

**Philosopher 2 is eating.**

**Philosopher 4 is hungry and trying to pick up forks.**

**Philosopher 4 picked up left fork.**

**Philosopher 0 put down right fork.**

**Philosopher 0 put down left fork.**

**Philosopher 0 is thinking.**

**Philosopher 4 picked up right fork.**

**Philosopher 4 is eating.**

**Philosopher 2 put down right fork.**

**Philosopher 2 put down left fork.**

**Philosopher 2 is thinking.**

**Philosopher 3 is hungry and trying to pick up forks.**

**Philosopher 3 picked up left fork.**

**Philosopher 1 is hungry and trying to pick up forks.**

**Philosopher 1 picked up left fork.**

**Philosopher 1 picked up right fork.**

**Philosopher 1 is eating.**

**Philosopher 4 put down right fork.**

**Philosopher 4 put down left fork.**

**Philosopher 4 is thinking.**

**Philosopher 3 picked up right fork.**

**Philosopher 3 is eating.**

**Philosopher 0 is hungry and trying to pick up forks.**

**Philosopher 0 picked up left fork.**

**Philosopher 2 is hungry and trying to pick up forks.**

**Philosopher 1 put down right fork.Philosopher 2 picked up left fork.**

**Philosopher 1 put down left fork.**

**Philosopher 1 is thinking.**

**Philosopher 0 picked up right fork.**

**Philosopher 0 is eating.**

**Philosopher 3 put down right fork.**

**Philosopher 3 put down left fork.**

**Philosopher 3 is thinking.**

**Philosopher 2 picked up right fork.**

**Philosopher 2 is eating.**

**Philosopher 4 is hungry and trying to pick up forks.**

**Philosopher 4 picked up left fork.**

**Philosopher 1 is hungry and trying to pick up forks.**

**Philosopher 2 put down right fork.**

**Philosopher 2 put down left fork.**

**Philosopher 2 is thinking.**

**Philosopher 3 is hungry and trying to pick up forks.**

**Philosopher 3 picked up left fork.**

**Philosopher 0 put down right fork.Philosopher 1 picked up left fork.**

**Philosopher 1 picked up right fork.**

**Philosopher 1 is eating.**

**Philosopher 0 put down left fork.**

**Philosopher 0 is thinking.**

**Philosopher 4 picked up right fork.**

**Philosopher 4 is eating.**

**Philosopher 1 put down right fork.**

**Philosopher 1 put down left fork.**

**Philosopher 1 is thinking.**

**Philosopher 4 put down right fork.**

**Philosopher 4 put down left fork.**

**Philosopher 4 is thinking.**

**Philosopher 3 picked up right fork.**

**Philosopher 3 is eating.**

**Philosopher 2 is hungry and trying to pick up forks.**

**Philosopher 2 picked up left fork.**

**Philosopher 0 is hungry and trying to pick up forks.**

**Philosopher 0 picked up left fork.**

**Philosopher 0 picked up right fork.**

**Philosopher 0 is eating.**

**Philosopher 1 is hungry and trying to pick up forks.**

**Philosopher 3 put down right fork.**

**Philosopher 3 put down left fork.**

**Philosopher 3 is thinking.**

**Philosopher 2 picked up right fork.**

**Philosopher 2 is eating.**

**Philosopher 4 is hungry and trying to pick up forks.**

**Philosopher 4 picked up left fork.**

**Philosopher 0 put down right fork.**

**Philosopher 0 put down left fork.**

**Philosopher 0 is thinking.**

**Philosopher 1 picked up left fork.**

**Philosopher 4 picked up right fork.**

**Philosopher 4 is eating.**

**Philosopher 2 put down right fork.**

**Philosopher 2 put down left fork.**

**Philosopher 2 is thinking.**

**Philosopher 1 picked up right fork.**

**Philosopher 1 is eating.**

**Philosopher 3 is hungry and trying to pick up forks.**

**Philosopher 3 picked up left fork.**

**Philosopher 4 put down right fork.**

**Philosopher 4 put down left fork.**

**Philosopher 4 is thinking.**

**Philosopher 3 picked up right fork.**

**Philosopher 3 is eating.**

**Philosopher 1 put down right fork.**

**Philosopher 1 put down left fork.**

**Philosopher 1 is thinking.**

**Philosopher 0 is hungry and trying to pick up forks.**

**Philosopher 0 picked up left fork.**

**Philosopher 0 picked up right fork.**

**Philosopher 0 is eating.**

**Philosopher 2 is hungry and trying to pick up forks.**

**Philosopher 2 picked up left fork.**

**Philosopher 1 is hungry and trying to pick up forks.**

**Philosopher 3 put down right fork.**

**Philosopher 3 put down left fork.**

**Philosopher 3 is thinking.**

**Philosopher 2 picked up right fork.**

**Philosopher 2 is eating.**

**Philosopher 4 is hungry and trying to pick up forks.**

**Philosopher 4 picked up left fork.**

**Philosopher 0 put down right fork.**

**Philosopher 0 put down left fork.**

**Philosopher 0 is thinking.**

**Philosopher 1 picked up left fork.**

**Philosopher 4 picked up right fork.**

**Philosopher 4 is eating.**

**Philosopher 3 is hungry and trying to pick up forks.**

**Philosopher 2 put down right fork.Philosopher 3 picked up left fork.**

**Philosopher 2 put down left fork.**

**Philosopher 2 is thinking.**

**Philosopher 1 picked up right fork.**

**Philosopher 1 is eating.**

**Philosopher 4 put down right fork.**

**Philosopher 4 put down left fork.**

**Philosopher 4 is thinking.**

**Philosopher 3 picked up right fork.**

**Philosopher 3 is eating.**

**Philosopher 1 put down right fork.**

**Philosopher 1 put down left fork.**

**Philosopher 1 is thinking.**

**Philosopher 0 is hungry and trying to pick up forks.**

**Philosopher 0 picked up left fork.**

**Philosopher 0 picked up right fork.**

**Philosopher 0 is eating.**

**Philosopher 2 is hungry and trying to pick up forks.**

**Philosopher 2 picked up left fork.**

**Philosopher 0 put down right fork.**

**Philosopher 0 put down left fork.**

**Philosopher 0 is thinking.**

**Philosopher 4 is hungry and trying to pick up forks.**

**Philosopher 3 put down right fork.Philosopher 4 picked up left fork.**

**Philosopher 4 picked up right fork.**

**Philosopher 4 is eating.**

**Philosopher 3 put down left fork.**

**Philosopher 3 is thinking.**

**Philosopher 2 picked up right fork.**

**Philosopher 2 is eating.**

**Philosopher 1 is hungry and trying to pick up forks.**

**Philosopher 1 picked up left fork.**

**Philosopher 0 is hungry and trying to pick up forks.**

**Philosopher 4 put down right fork.**

**Philosopher 4 put down left fork.**

**Philosopher 4 is thinking.**

**Philosopher 0 picked up left fork.**

**Philosopher 2 put down right fork.**

**Philosopher 2 put down left fork.**

**Philosopher 2 is thinking.**

**Philosopher 1 picked up right fork.**

**Philosopher 1 is eating.**

**Philosopher 3 is hungry and trying to pick up forks.**

**Philosopher 3 picked up left fork.**

**Philosopher 3 picked up right fork.**

**Philosopher 3 is eating.**

**Philosopher 1 put down right fork.**

**Philosopher 1 put down left fork.**

**Philosopher 1 is thinking.**

**Philosopher 0 picked up right fork.**

**Philosopher 0 is eating.**

**Philosopher 4 is hungry and trying to pick up forks.**

**Philosopher 2 is hungry and trying to pick up forks.**

**Philosopher 2 picked up left fork.**

**Philosopher 3 put down right fork.Philosopher 4 picked up left fork.**

**Philosopher 3 put down left fork.**

**Philosopher 3 is thinking.**

**Philosopher 2 picked up right fork.**

**Philosopher 2 is eating.**

**Philosopher 0 put down right fork.**

**Philosopher 0 put down left fork.**

**Philosopher 0 is thinking.**

**Philosopher 4 picked up right fork.**

**Philosopher 4 is eating.**

**Philosopher 2 put down right fork.**

**Philosopher 2 put down left fork.**

**Philosopher 2 is thinking.**

**Philosopher 3 is hungry and trying to pick up forks.**

**Philosopher 3 picked up left fork.**

**Philosopher 1 is hungry and trying to pick up forks.**

**Philosopher 1 picked up left fork.**

**Philosopher 1 picked up right fork.**

**Philosopher 1 is eating.**

**Philosopher 0 is hungry and trying to pick up forks.**

**Philosopher 4 put down right fork.**

**Philosopher 4 put down left fork.**

**Philosopher 4 is thinking.**

**Philosopher 0 picked up left fork.**

**Philosopher 3 picked up right fork.**

**Philosopher 3 is eating.**

**Philosopher 1 put down right fork.**

**Philosopher 1 put down left fork.**

**Philosopher 1 is thinking.**

**Philosopher 0 picked up right fork.**

**Philosopher 0 is eating.**

**Philosopher 4 is hungry and trying to pick up forks.**

**Philosopher 3 put down right fork.**

**Philosopher 3 put down left fork.**

**Philosopher 3 is thinking.**

**Philosopher 4 picked up left fork.**

**Philosopher 2 is hungry and trying to pick up forks.**

**Philosopher 2 picked up left fork.**

**Philosopher 2 picked up right fork.**

**Philosopher 2 is eating.**

**Philosopher 0 put down right fork.**

**Philosopher 0 put down left fork.**

**Philosopher 0 is thinking.**

**Philosopher 4 picked up right fork.**

**Philosopher 4 is eating.**

**Philosopher 1 is hungry and trying to pick up forks.**

**Philosopher 1 picked up left fork.**

**Philosopher 3 is hungry and trying to pick up forks.**

**Philosopher 2 put down right fork.**

**Philosopher 2 put down left fork.**

**Philosopher 2 is thinking.**

**Philosopher 3 picked up left fork.**

**Philosopher 1 picked up right fork.**

**Philosopher 1 is eating.**

**Philosopher 4 put down right fork.**

**Philosopher 4 put down left fork.**

**Philosopher 4 is thinking.**

**Philosopher 3 picked up right fork.**

**Philosopher 3 is eating.**

**Philosopher 0 is hungry and trying to pick up forks.**

**Philosopher 0 picked up left fork.**

**Philosopher 2 is hungry and trying to pick up forks.**

**Philosopher 3 put down right fork.**

**Philosopher 3 put down left fork.**

**Philosopher 3 is thinking.**

**Philosopher 1 put down right fork.Philosopher 2 picked up left fork.**

**Philosopher 2 picked up right fork.**

**Philosopher 2 is eating.**

**Philosopher 1 put down left fork.**

**Philosopher 1 is thinking.**

**Philosopher 0 picked up right fork.**

**Philosopher 0 is eating.**

**Philosopher 4 is hungry and trying to pick up forks.**

**Philosopher 4 picked up left fork.**

**Philosopher 0 put down right fork.**

**Philosopher 0 put down left fork.**

**Philosopher 0 is thinking.**

**Philosopher 4 picked up right fork.**

**Philosopher 4 is eating.**

**Philosopher 3 is hungry and trying to pick up forks.**

**Philosopher 2 put down right fork.**

**Philosopher 2 put down left fork.**

**Philosopher 2 is thinking.**

**Philosopher 3 picked up left fork.**

**Philosopher 1 is hungry and trying to pick up forks.**

**Philosopher 1 picked up left fork.**

**Philosopher 1 picked up right fork.**

**Philosopher 1 is eating.**

**Philosopher 4 put down right fork.**

**Philosopher 4 put down left fork.**

**Philosopher 4 is thinking.**

**Philosopher 3 picked up right fork.**

**Philosopher 3 is eating.**

**Philosopher 2 is hungry and trying to pick up forks.**

**Philosopher 0 is hungry and trying to pick up forks.**

**Philosopher 0 picked up left fork.**

**Philosopher 1 put down right fork.**

**Philosopher 1 put down left fork.**

**Philosopher 1 is thinking.**

**Philosopher 2 picked up left fork.**

**Philosopher 0 picked up right fork.**

**Philosopher 0 is eating.**

**Philosopher 3 put down right fork.**

**Philosopher 3 put down left fork.**

**Philosopher 3 is thinking.**

**Philosopher 2 picked up right fork.**

**Philosopher 2 is eating.**

**Philosopher 1 is hungry and trying to pick up forks.**

**Philosopher 4 is hungry and trying to pick up forks.**

**Philosopher 4 picked up left fork.**

**Philosopher 0 put down right fork.**

**Philosopher 0 put down left fork.**

**Philosopher 0 is thinking.**

**Philosopher 1 picked up left fork.**

**Philosopher 4 picked up right fork.**

**Philosopher 4 is eating.**

**Philosopher 2 put down right fork.**

**Philosopher 2 put down left fork.**

**Philosopher 2 is thinking.**

**Philosopher 1 picked up right fork.**

**Philosopher 1 is eating.**

**Philosopher 3 is hungry and trying to pick up forks.**

**Philosopher 3 picked up left fork.**

**Philosopher 4 put down right fork.**

**Philosopher 4 put down left fork.**

**Philosopher 4 is thinking.**

**Philosopher 3 picked up right fork.**

**Philosopher 3 is eating.**

**Philosopher 1 put down right fork.**

**Philosopher 1 put down left fork.**

**Philosopher 1 is thinking.**

**Philosopher 4 is hungry and trying to pick up forks.**

**Philosopher 0 is hungry and trying to pick up forks.**

**Philosopher 0 picked up left fork.**

**Philosopher 0 picked up right fork.**

**Philosopher 0 is eating.**

**Philosopher 3 put down right fork.Philosopher 4 picked up left fork.**

**Philosopher 3 put down left fork.**

**Philosopher 3 is thinking.**

**Philosopher 2 is hungry and trying to pick up forks.**

**Philosopher 2 picked up left fork.**

**Philosopher 2 picked up right fork.**

**Philosopher 2 is eating.**

**Philosopher 2 put down right fork.**

**Philosopher 2 put down left fork.**

**Philosopher 2 is thinking.**

**Philosopher 0 put down right fork.**

**Philosopher 0 put down left fork.**

**Philosopher 0 is thinking.**

**Philosopher 4 picked up right fork.**

**Philosopher 4 is eating.**

**Philosopher 1 is hungry and trying to pick up forks.**

**Philosopher 1 picked up left fork.**

**Philosopher 1 picked up right fork.**

**Philosopher 1 is eating.**

**Philosopher 3 is hungry and trying to pick up forks.**

**Philosopher 3 picked up left fork.**

**Philosopher 2 is hungry and trying to pick up forks.**

**Philosopher 0 is hungry and trying to pick up forks.**

**In [ ]:**

**Experiment11:**

**​**# Banker's Algorithm in Python

def is\_safe(processes, available, max\_matrix, allocation, need):

"""Check if the system is in a safe state."""

work = available[:]

finish = [False] \* len(processes)

safe\_seq = []

while len(safe\_seq) < len(processes):

found = False

for p in range(len(processes)):

if not finish[p]:

if all(need[p][r] <= work[r] for r in range(len(available))):

for r in range(len(available)):

work[r] += allocation[p][r]

safe\_seq.append(p)

finish[p] = True

found = True

break

if not found:

return False, [] # System is not in a safe state

return True, safe\_seq

def request\_resources(process\_id, request, available, allocation, need):

"""Attempt to allocate requested resources to the process."""

if all(request[r] <= need[process\_id][r] for r in range(len(request))):

if all(request[r] <= available[r] for r in range(len(request))):

# Pretend to allocate the resources

for r in range(len(request)):

available[r] -= request[r]

allocation[process\_id][r] += request[r]

need[process\_id][r] -= request[r]

# Check if system is safe

safe, \_ = is\_safe(processes, available, max\_matrix, allocation, need)

if safe:

return True # Allocation is safe

else:

# Roll back the allocation

for r in range(len(request)):

available[r] += request[r]

allocation[process\_id][r] -= request[r]

need[process\_id][r] += request[r]

return False # System would be unsafe

else:

print("The resources are not available.")

return False

else:

print("Request exceeds maximum claim.")

return False

# Number of processes and resources

processes = [0, 1, 2, 3, 4]

num\_resources = 3

# Available resources

available = [3, 3, 2]

# Maximum resources required by each process

max\_matrix = [

[7, 5, 3], # P0

[3, 2, 2], # P1

[9, 0, 2], # P2

[2, 2, 2], # P3

[4, 3, 3] # P4

]

# Resources currently allocated to each process

allocation = [

[0, 1, 0], # P0

[2, 0, 0], # P1

[3, 0, 2], # P2

[2, 1, 1], # P3

[0, 0, 2] # P4

]

# Resources still needed by each process (calculated as Max - Allocation)

need = [

[7, 4, 3], # P0

[1, 2, 2], # P1

[6, 0, 0], # P2

[0, 1, 1], # P3

[4, 3, 1] # P4

]

# Test request

request = [1, 0, 2]

process\_id = 1

if request\_resources(process\_id, request, available, allocation, need):

print("Resources allocated successfully.")

else:

print("Request denied due to unsafe state.")

# Check system state

safe, safe\_seq = is\_safe(processes, available, max\_matrix, allocation, need)

if safe:

print(f"System is in a safe state. Safe sequence: {safe\_seq}")

else:

print("System is not in a safe state.")

Output

Resources allocated successfully.

System is in a safe state. Safe sequence: [1, 3, 0, 2, 4]