```
th the α Share Run
                                                                                                                                                                                                                                                                                                                                                                                                      Clear
                                                                                                                                                                                                              Output
  main.c
                                                                                                                                                                                                              Thread 1: 1
  #include opthread.h>
#include stdib.h>
#include sunistd.h>
*void* print_numbers(void* arg) {
                                                                                                                                                                                                              Thread 2: A
Thread 1: 2
                                                                                                                                                                                                              Thread 2: 8
Thread 1: 3
5 · void* print_numbers(void* arg) {
6     int i;
7     for (i = 1; i <= 5; i++) {
8         printf("Thread 1: %d\n", i);
9         sleep(1);}
10     return NULL;}
11 · void* print_letters(void* arg) {
12         char ch:
                                                                                                                                                                                                             Thread 2: C
Thread 1: 4
                                                                                                                                                                                                             Thread 2: 0
Thread 1: 5
                                                                                                                                                                                                             Thread 2: E
Both threads finished execution
               char ch;
for (ch = 'A'; ch <= 'E'; ch++) {
    printf("Thread 2: %c\n", ch);
    sleep(1);}</pre>
 12
 13 •
  14
                                                                                                                                                                                                              === Code Execution Successful ===
 16 return NULL:}
17- int main() {
                 pthread_t thread1, thread2;
if (pthread_create(&thread1, NULL, print_numbers, NULL)) {
 18
19 •
 20
21
                       fprintf(stderr, "Error creating thread 1\n");
                         return 1:}
 22 -
                 if (pthread_create(&thread2, NULL, print_letters, NULL)) {
                fprintf(stder, "Error creating thread 2\n");
return 1;

if (pthread_join(thread1, NULL)) {
    fprintf(stderr, "Error joining thread 1\n");
    return 1;
 23
 24
25 -
26
27
28 -
29
30
31
                return 1;)
if (pthread_join(thread2, NMLL)) {
    fprintf(stderr, "Error joining thread 2\n");
    return 1;}
printf("Both threads finished execution\n");
32
```

```
main.c
                                                                                                                                                Output
 1 #include <stdio.h>
                                                                                                                                               Philosopher 0 is thinking.
2 #include <stdlib.h>
                                                                                                                                               Philosopher 1 is thinking.
3 #include opthread.ho
                                                                                                                                               Philosopher 2 is thinking.
 4 #include <unistd.h>
                                                                                                                                               Philosopher 3 is thinking.
5 #define NUM_PHILOSOPHERS 4
                                                                                                                                               Philosopher 2 is eating.
 6 #define NUM EATS 2
                                                                                                                                               Philosopher 0 is eating.
 7 pthread_mutex_t forks[NUM_PHILOSOPHERS]:
                                                                                                                                               Philosopher 0 is thinking.
 8 - void think(int philosopher) {
                                                                                                                                               Philosopher 2 is thinking.
        printf("Philosopher %d is thinking.\n", philosopher);
                                                                                                                                               Philosopher 1 is eating.
        sleep(rand() % 3 + 1);}
                                                                                                                                               Philosopher 3 is eating.
10
                                                                                                                                               Philosopher 1 is thinking.
11 - void eat(int philosopher) {
       printf("Philosopher %d is eating.\n", philosopher);
                                                                                                                                               Philosopher 3 is thinking.
13
        sleep(rand() % 3 + 1);}
                                                                                                                                               Philosopher 0 is eating.
                                                                                                                                               Philosopher 2 is eating.
14 - void* philosopher_action(void* arg) {
15
       int philosopher - "((int*)arg);
                                                                                                                                               Philosopher 3 is eating.
16 .
        for (int 1 - 0; 1 < NUM_EATS; 1++) {
                                                                                                                                               Philosopher 1 is eating.
17
           think(philosopher);
18
            pthread_mutex_lock(&forks[philosopher]);
19
            pthread_nutex_lock(&forks[(philosopher + 1) % NUM_PHILOSOPHERS]);
                                                                                                                                               --- Code Execution Successful ---
20
          eat(philosopher);
            pthread_mutex_unlock(&forks[(philosopher - 1) % NUM_PHILOSOPHERS]);
21
           pthread_mutex_unlock(&forks[philosopher]);}
22
23
        return NULLIF
24 - int main() {
25
       pthread_t threads[NUM_PHILOSOPHERS];
26
        int philosophers[NUM_PHILOSOPHERS];
27.
        for (int i - 0; i < NUM_PHILOSOPHERS; i++) {
28
           pthread_mutex_init(&forks[i], NULL);}
29 .
        for (int 1 - 0; 1 < NUN_PHILOSOPHERS; 1\leftrightarrow) {
30
          philosophers[i] - 1;
31 .
            if (pthread_create(&threads[i], NULL, philosopher_action, &philosophers[i]) |- 0) {
            perror("Failed to create thread");
32
33
               return 1;}}
        for (int 1 - 0: 1 < NUM PHILOSOPHERS: 1++) [
34 +
35
          pthread_join(threads[i], NULL);}
36+
        for (int 1 - 0; 1 < NUM_PHILOSOPHERS; 1++) {
37
           pthread_mutex_destroy(&forks[i]);}
       return 0;}
```