# PES UNIVERSITY

**Department of Computer Science & Engineering**

## Microprocessor & Computer Architecture Lab

## UE23CS251B

## MiniProject Coding Assignment

| | |
|---|---|
| Name of the Student | Pranav Hemanth |
| SRN | PES1UG23CS433 |
| Section | G |
| Department | CSE |
| Campus | RR |

| | |
|---|---|
| Name of the Student | Nishant K Holla |
| SRN | PES1UG23CS401 |
| Section | G |
| Department | CSE |
| Campus | RR |

| | |
|---|---|
| Name of the Student | Nishant Jayaram Hegde |
| SRN | PES1UG23CS400 |
| Section | G |

Jan -May 2025 LAB SUBMISSION_**UE23CS251B**

| | |
|---|---|
| **Department** | **CSE** |
| **Campus** | **RR** |

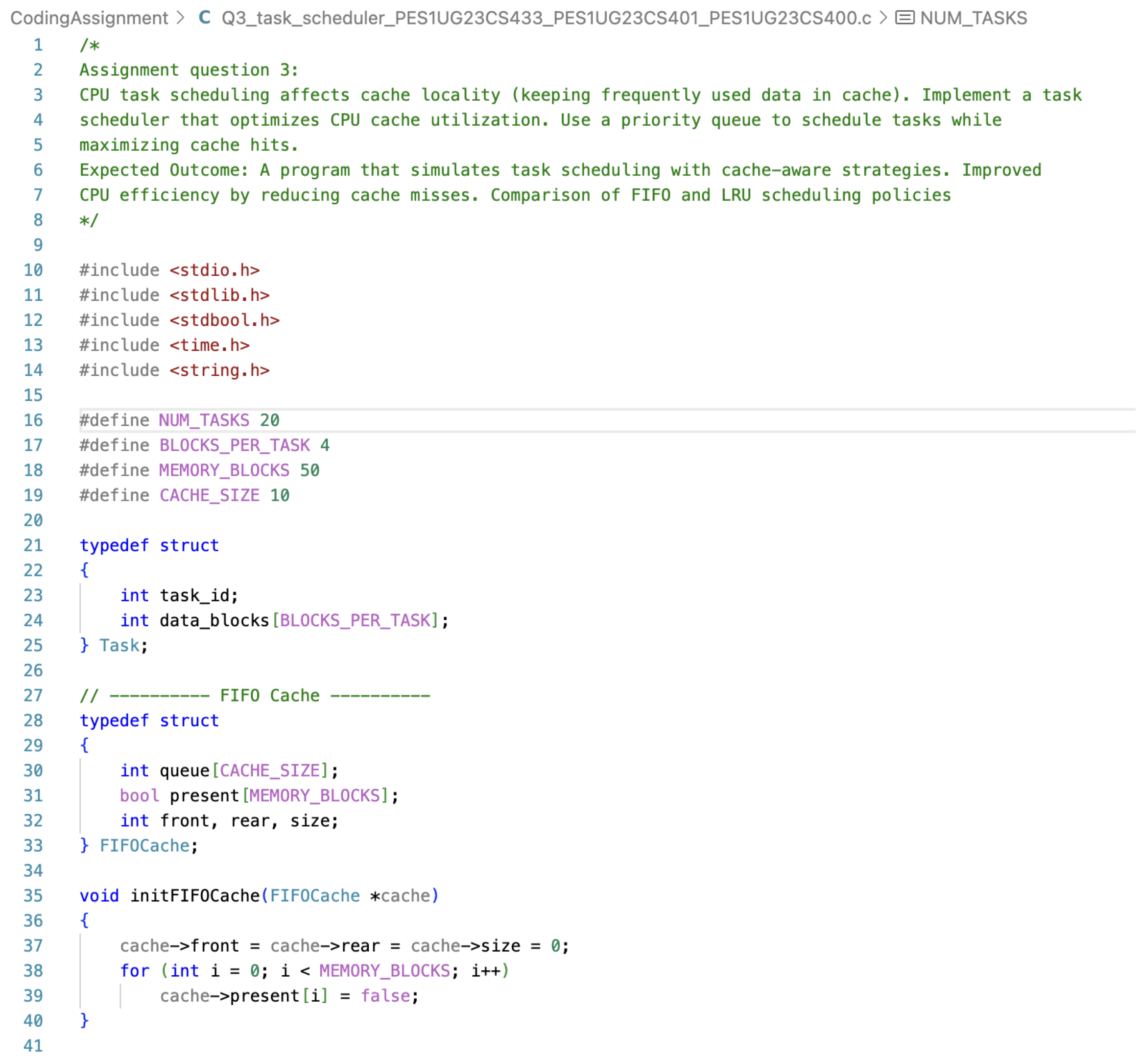
**Department of Computer Science & Engineering**
**Microprocessor & Computer Architecture Lab**

**UE23CS251B**

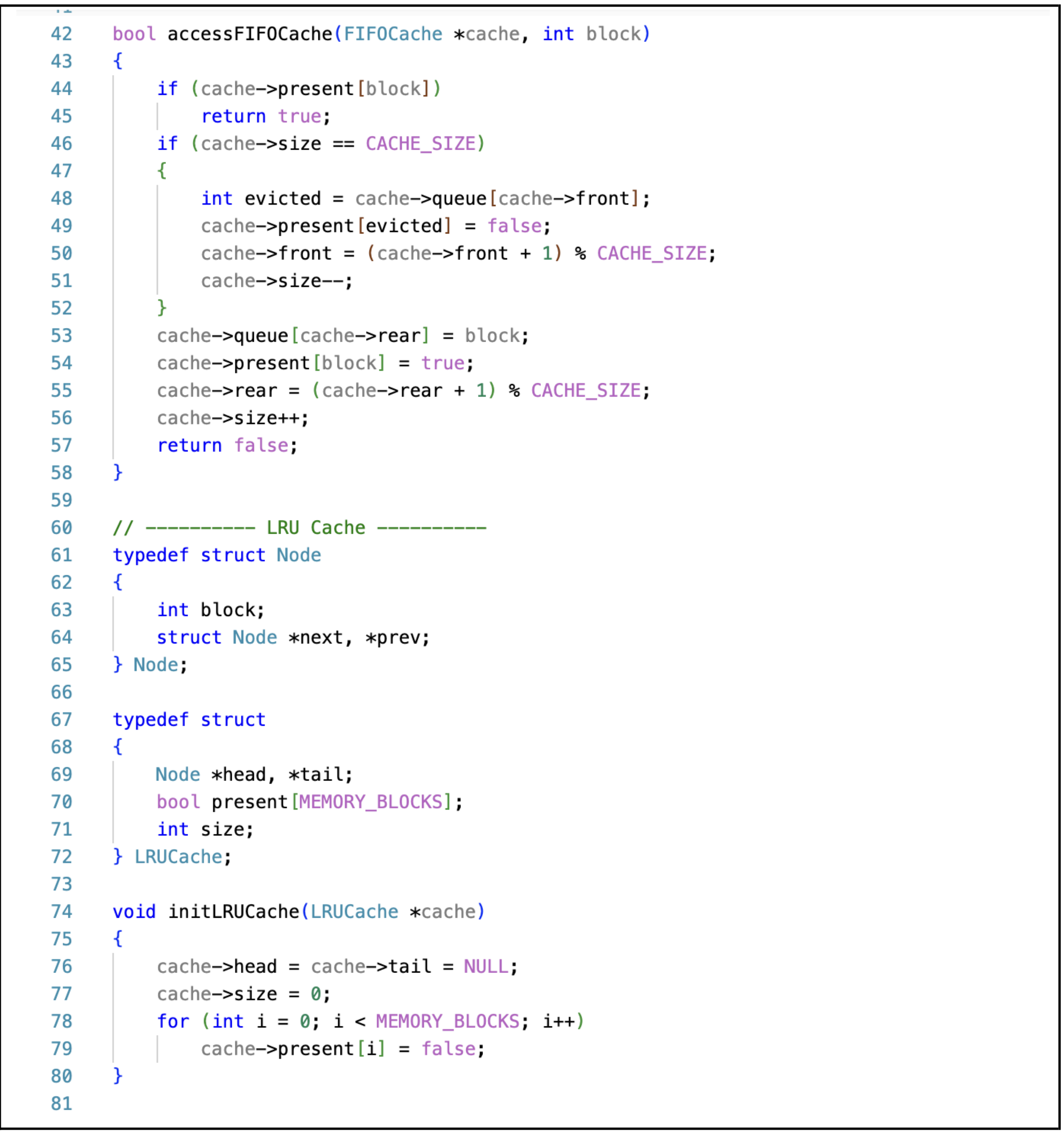
Q3

Assignment question 3:
CPU task scheduling affects cache locality (keeping frequently used data in cache). Implement a task scheduler that optimizes CPU cache utilization. Use a priority queue to schedule tasks while maximizing cache hits.
Expected Outcome: A program that simulates task scheduling with cache-aware strategies. Improved CPU efficiency by reducing cache misses. Comparison of FIFO and LRU scheduling policies

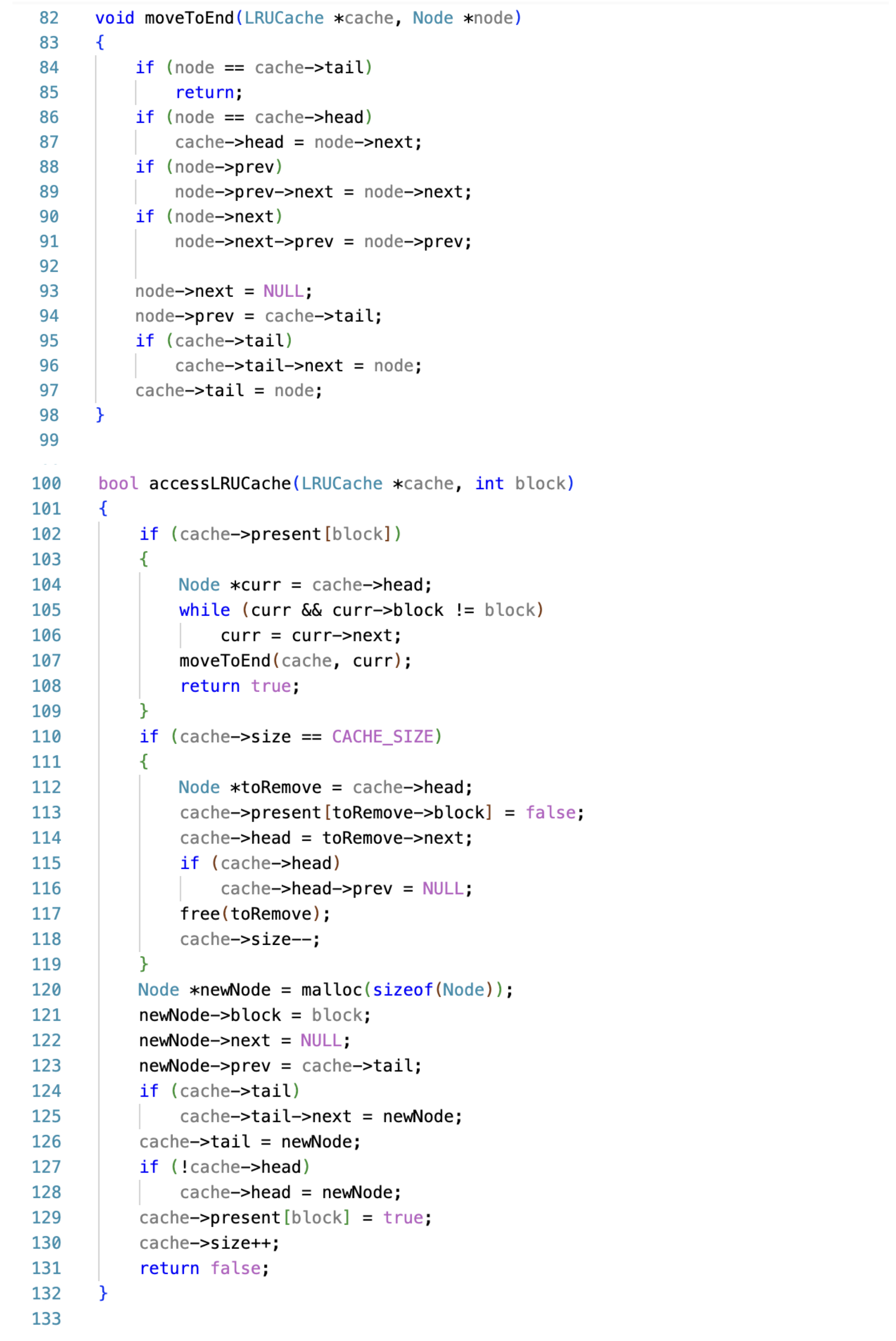
Code:


CodingAssignment > C Q3_task_scheduler_PES1UG23CS433_PES1UG23CS401_PES1UG23CS400.c > NUM_TASKS

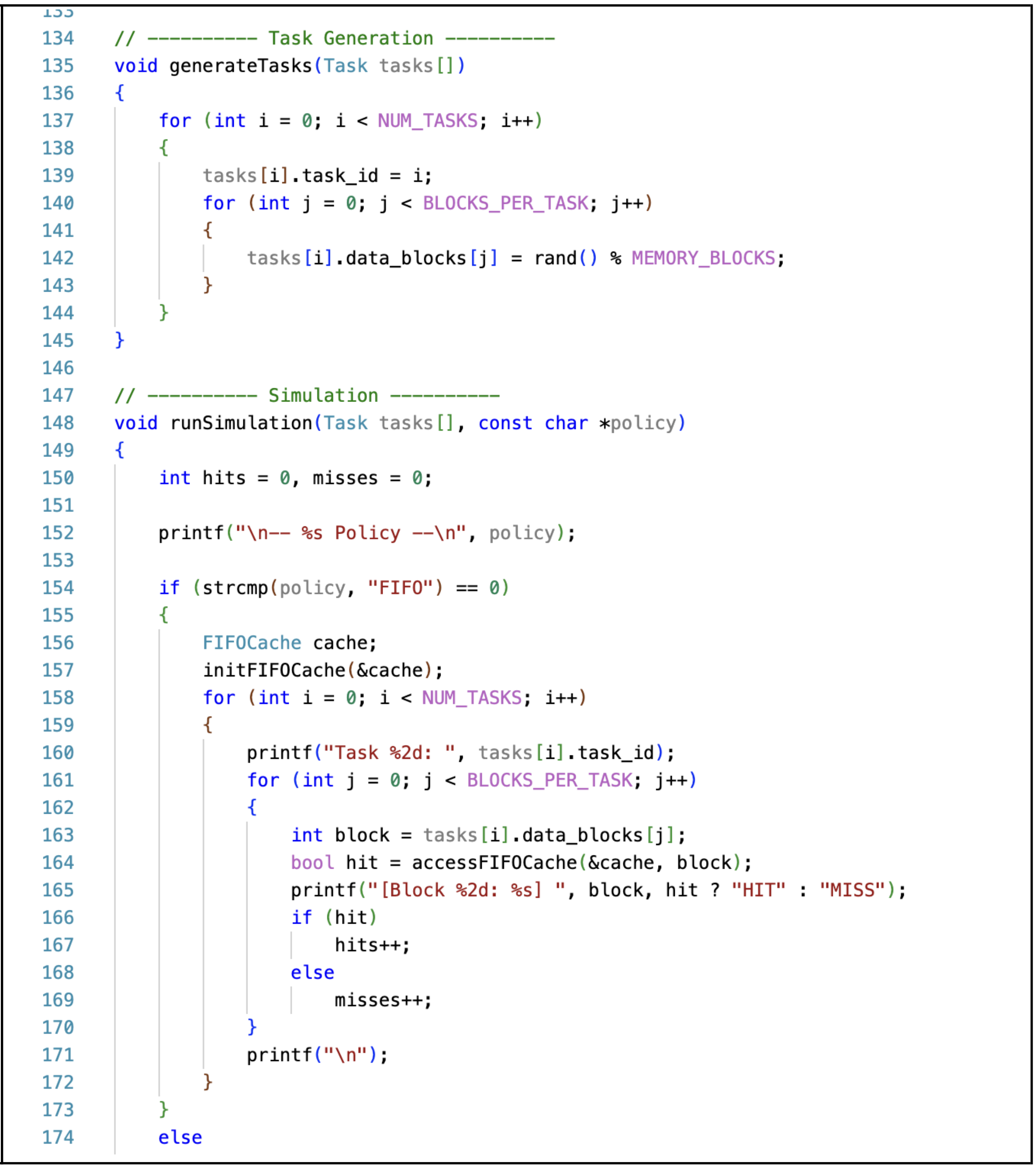
```c
/*
Assignment question 3:
CPU task scheduling affects cache locality (keeping frequently used data in cache). Implement a task
scheduler that optimizes CPU cache utilization. Use a priority queue to schedule tasks while
maximizing cache hits.
Expected Outcome: A program that simulates task scheduling with cache-aware strategies. Improved
CPU efficiency by reducing cache misses. Comparison of FIFO and LRU scheduling policies
*/

#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>
#include <string.h>

#define NUM_TASKS 20
#define BLOCKS_PER_TASK 4
#define MEMORY_BLOCKS 50
#define CACHE_SIZE 10

typedef struct
{
    int task_id;
    int data_blocks[BLOCKS_PER_TASK];
} Task;

// ---------- FIFO Cache ----------
typedef struct
{
    int queue[CACHE_SIZE];
    bool present[MEMORY_BLOCKS];
    int front, rear, size;
} FIFOCache;

void initFIFOCache(FIFOCache *cache)
{
    cache->front = cache->rear = cache->size = 0;
    for (int i = 0; i < MEMORY_BLOCKS; i++)
        cache->present[i] = false;
}

```



```
bool accessFIFOCache(FIFOCache *cache, int block)
{
    if (cache->present[block])
        return true;
    if (cache->size == CACHE_SIZE)
    {
        int evicted = cache->queue[cache->front];
        cache->present[evicted] = false;
        cache->front = (cache->front + 1) % CACHE_SIZE;
        cache->size--;
    }
    cache->queue[cache->rear] = block;
    cache->present[block] = true;
    cache->rear = (cache->rear + 1) % CACHE_SIZE;
    cache->size++;
    return false;
}

// ---------- LRU Cache ----------
typedef struct Node
{
    int block;
    struct Node *next, *prev;
} Node;

typedef struct
{
    Node *head, *tail;
    bool present[MEMORY_BLOCKS];
    int size;
} LRUCache;

void initLRUCache(LRUCache *cache)
{
    cache->head = cache->tail = NULL;
    cache->size = 0;
    for (int i = 0; i < MEMORY_BLOCKS; i++)
        cache->present[i] = false;
}

```

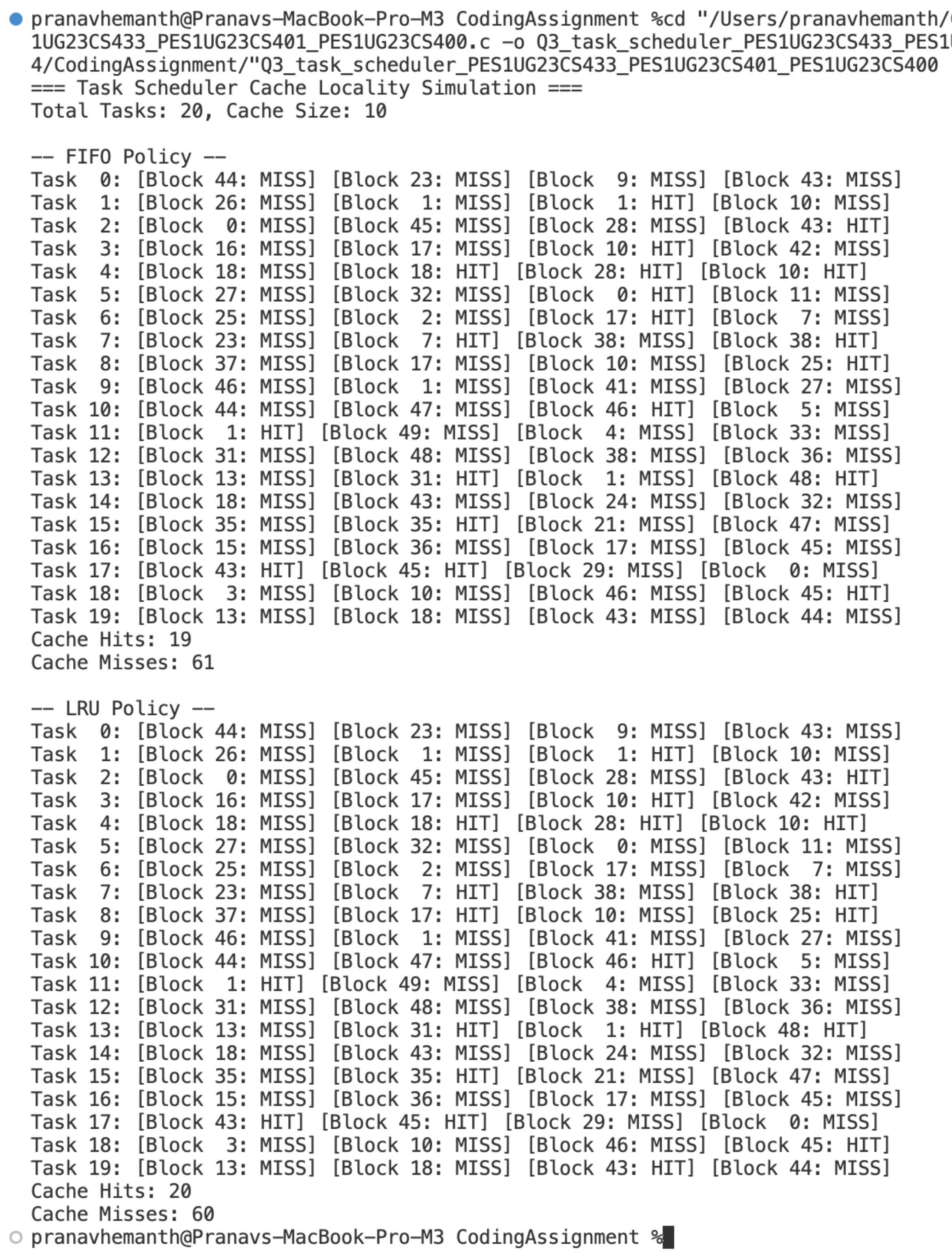


```c
void moveToEnd(LRUCache *cache, Node *node)
{
    if (node == cache->tail)
        return;
    if (node == cache->head)
        cache->head = node->next;
    if (node->prev)
        node->prev->next = node->next;
    if (node->next)
        node->next->prev = node->prev;

    node->next = NULL;
    node->prev = cache->tail;
    if (cache->tail)
        cache->tail->next = node;
    cache->tail = node;
}

bool accessLRUCache(LRUCache *cache, int block)
{
    if (cache->present[block])
    {
        Node *curr = cache->head;
        while (curr && curr->block != block)
            curr = curr->next;
        moveToEnd(cache, curr);
        return true;
    }
    if (cache->size == CACHE_SIZE)
    {
        Node *toRemove = cache->head;
        cache->present[toRemove->block] = false;
        cache->head = toRemove->next;
        if (cache->head)
            cache->head->prev = NULL;
        free(toRemove);
        cache->size--;
    }
    Node *newNode = malloc(sizeof(Node));
    newNode->block = block;
    newNode->next = NULL;
    newNode->prev = cache->tail;
    if (cache->tail)
        cache->tail->next = newNode;
    cache->tail = newNode;
    if (!cache->head)
        cache->head = newNode;
    cache->present[block] = true;
    cache->size++;
    return false;
}

```



```c

// ---------- Task Generation ----------
void generateTasks(Task tasks[])
{
    for (int i = 0; i < NUM_TASKS; i++)
    {
        tasks[i].task_id = i;
        for (int j = 0; j < BLOCKS_PER_TASK; j++)
        {
            tasks[i].data_blocks[j] = rand() % MEMORY_BLOCKS;
        }
    }
}

// ---------- Simulation ----------
void runSimulation(Task tasks[], const char *policy)
{
    int hits = 0, misses = 0;

    printf("\n-- %s Policy --\n", policy);

    if (strcmp(policy, "FIFO") == 0)
    {
        FIFOCache cache;
        initFIFOCache(&cache);
        for (int i = 0; i < NUM_TASKS; i++)
        {
            printf("Task %2d: ", tasks[i].task_id);
            for (int j = 0; j < BLOCKS_PER_TASK; j++)
            {
                int block = tasks[i].data_blocks[j];
                bool hit = accessFIFOCache(&cache, block);
                printf("[Block %2d: %s] ", block, hit ? "HIT" : "MISS");
                if (hit)
                    hits++;
                else
                    misses++;
            }
            printf("\n");
        }
    }
    else
    else
    {
        LRUCache cache;
        initLRUCache(&cache);
        for (int i = 0; i < NUM_TASKS; i++)
        {
            printf("Task %2d: ", tasks[i].task_id);
            for (int j = 0; j < BLOCKS_PER_TASK; j++)
            {
                int block = tasks[i].data_blocks[j];
                bool hit = accessLRUCache(&cache, block);
                printf("[Block %2d: %s] ", block, hit ? "HIT" : "MISS");
                if (hit)
                    hits++;
                else
                    misses++;
            }
            printf("\n");
        }
    }

    printf("Cache Hits: %d\n", hits);
    printf("Cache Misses: %d\n", misses);
}

// ---------- Main ----------
int main()
{
    srand(42);
    Task tasks[NUM_TASKS];
    generateTasks(tasks);

    printf("=== Task Scheduler Cache Locality Simulation ===\n");
    printf("Total Tasks: %d, Cache Size: %d\n", NUM_TASKS, CACHE_SIZE);

    runSimulation(tasks, "FIFO");
    runSimulation(tasks, "LRU");

    return 0;
}
```

Screenshot:



```
● pranavhemanth@Pranavs-MacBook-Pro-M3 CodingAssignment %cd "/Users/pranavhemanth/(
  1UG23CS433_PES1UG23CS401_PES1UG23CS400.c -o Q3_task_scheduler_PES1UG23CS433_PES1(
  4/CodingAssignment/"Q3_task_scheduler_PES1UG23CS433_PES1UG23CS401_PES1UG23CS400
  === Task Scheduler Cache Locality Simulation ===
  Total Tasks: 20, Cache Size: 10

  -- FIFO Policy --
  Task  0: [Block 44: MISS] [Block 23: MISS] [Block  9: MISS] [Block 43: MISS]
  Task  1: [Block 26: MISS] [Block  1: MISS] [Block  1: HIT] [Block 10: MISS]
  Task  2: [Block  0: MISS] [Block 45: MISS] [Block 28: MISS] [Block 43: HIT]
  Task  3: [Block 16: MISS] [Block 17: MISS] [Block 10: HIT] [Block 42: MISS]
  Task  4: [Block 18: MISS] [Block 18: HIT] [Block 28: HIT] [Block 10: HIT]
  Task  5: [Block 27: MISS] [Block 32: MISS] [Block  0: HIT] [Block 11: MISS]
  Task  6: [Block 25: MISS] [Block  2: MISS] [Block 17: HIT] [Block  7: MISS]
  Task  7: [Block 23: MISS] [Block  7: HIT] [Block 38: MISS] [Block 38: HIT]
  Task  8: [Block 37: MISS] [Block 17: MISS] [Block 10: MISS] [Block 25: HIT]
  Task  9: [Block 46: MISS] [Block  1: MISS] [Block 41: MISS] [Block 27: MISS]
  Task 10: [Block 44: MISS] [Block 47: MISS] [Block 46: HIT] [Block  5: MISS]
  Task 11: [Block  1: HIT] [Block 49: MISS] [Block  4: MISS] [Block 33: MISS]
  Task 12: [Block 31: MISS] [Block 48: MISS] [Block 38: MISS] [Block 36: MISS]
  Task 13: [Block 13: MISS] [Block 31: HIT] [Block  1: MISS] [Block 48: HIT]
  Task 14: [Block 18: MISS] [Block 43: MISS] [Block 24: MISS] [Block 32: MISS]
  Task 15: [Block 35: MISS] [Block 35: HIT] [Block 21: MISS] [Block 47: MISS]
  Task 16: [Block 15: MISS] [Block 36: MISS] [Block 17: MISS] [Block 45: MISS]
  Task 17: [Block 43: HIT] [Block 45: HIT] [Block 29: MISS] [Block  0: MISS]
  Task 18: [Block  3: MISS] [Block 10: MISS] [Block 46: MISS] [Block 45: HIT]
  Task 19: [Block 13: MISS] [Block 18: MISS] [Block 43: MISS] [Block 44: MISS]
  Cache Hits: 19
  Cache Misses: 61

  -- LRU Policy --
  Task  0: [Block 44: MISS] [Block 23: MISS] [Block  9: MISS] [Block 43: MISS]
  Task  1: [Block 26: MISS] [Block  1: MISS] [Block  1: HIT] [Block 10: MISS]
  Task  2: [Block  0: MISS] [Block 45: MISS] [Block 28: MISS] [Block 43: HIT]
  Task  3: [Block 16: MISS] [Block 17: MISS] [Block 10: HIT] [Block 42: MISS]
  Task  4: [Block 18: MISS] [Block 18: HIT] [Block 28: HIT] [Block 10: HIT]
  Task  5: [Block 27: MISS] [Block 32: MISS] [Block  0: MISS] [Block 11: MISS]
  Task  6: [Block 25: MISS] [Block  2: MISS] [Block 17: MISS] [Block  7: MISS]
  Task  7: [Block 23: MISS] [Block  7: HIT] [Block 38: MISS] [Block 38: HIT]
  Task  8: [Block 37: MISS] [Block 17: HIT] [Block 10: MISS] [Block 25: HIT]
  Task  9: [Block 46: MISS] [Block  1: MISS] [Block 41: MISS] [Block 27: MISS]
  Task 10: [Block 44: MISS] [Block 47: MISS] [Block 46: HIT] [Block  5: MISS]
  Task 11: [Block  1: HIT] [Block 49: MISS] [Block  4: MISS] [Block 33: MISS]
  Task 12: [Block 31: MISS] [Block 48: MISS] [Block 38: MISS] [Block 36: MISS]
  Task 13: [Block 13: MISS] [Block 31: HIT] [Block  1: HIT] [Block 48: HIT]
  Task 14: [Block 18: MISS] [Block 43: MISS] [Block 24: MISS] [Block 32: MISS]
  Task 15: [Block 35: MISS] [Block 35: HIT] [Block 21: MISS] [Block 47: MISS]
  Task 16: [Block 15: MISS] [Block 36: MISS] [Block 17: MISS] [Block 45: MISS]
  Task 17: [Block 43: HIT] [Block 45: HIT] [Block 29: MISS] [Block  0: MISS]
  Task 18: [Block  3: MISS] [Block 10: MISS] [Block 46: MISS] [Block 45: HIT]
  Task 19: [Block 13: MISS] [Block 18: MISS] [Block 43: HIT] [Block 44: MISS]
  Cache Hits: 20
  Cache Misses: 60
○ pranavhemanth@Pranavs-MacBook-Pro-M3 CodingAssignment %
```