Question 2: Parallel Max Pooling Implementation

This C program implements a parallel max pooling operation using shared memory for inter-process communication. Max pooling is commonly used in convolutional neural networks to reduce the spatial dimensions of data.

Core Components

1. Program Configuration

```
#define M 5    // Input matrix height
#define N 5    // Input matrix width
#define K 2    // Pooling window height
#define L 2    // Pooling window width
```

Output dimensions are calculated as:

```
#define M_OUT ((M + K - 1) / K)  // Output matrix height
#define N_OUT ((N + L - 1) / L)  // Output matrix width
```

2. Shared Memory Structure

The program uses POSIX shared memory for inter-process communication:

```
typedef struct {
    int result[M_OUT][N_OUT];
} SharedData;
```

3. Key Functions

Max Pooling Operation

`max_pool()` function:

- Takes input matrix and starting position
- Finds maximum value in KxL window
- Includes detailed logging of the process
- Returns maximum value found in the window

Process Management

- Parent process:

    - Creates and initializes random input matrix
    - Sets up shared memory
    - Spawns child processes
    - Collects and displays results

- Child processes:

    - Each handles one row block of the input matrix
    - Compute max pooling results
    - Write results to shared memory

Shared Memory Communication

Setup

```
int shm_fd = shm_open(shm_name, O_CREAT | O_RDWR, 0666);
ftruncate(shm_fd, sizeof(SharedData));
SharedData *shared_data = mmap(NULL, sizeof(SharedData),
                               PROT_READ | PROT_WRITE,
                               MAP_SHARED, shm_fd, 0);
```

Usage

- Parent process creates shared memory segment
- Child processes write their results directly to shared memory
- Parent process reads final results after all children complete
- Shared memory is properly cleaned up using `munmap` and `shm_unlink`

Program Flow

1. Initialize random input matrix
2. Create shared memory segment
3. Fork child processes for each row block
4. Children compute max pooling results
5. Parent waits for all children to complete
6. Copy results from shared memory
7. Display output matrix
8. Clean up shared memory