## use cmd+shift+v or ctrl+shift+v to view markdown in vscode

## Matrix Multiplication Using Processes

You are given two matrices stored in two separate files. The first file (`matrix1.txt`) contains a matrix of size ( n \times m ), and the second file (`matrix2.txt`) contains a matrix of size ( m \times q ). Write a C program to perform matrix multiplication using processes, adhering to the following steps:

1. **Reading Matrices:**

   - In the `main` function, read the contents of `matrix1.txt` into a 2D array representing Matrix A of size ( n \times m ).
   - Read the contents of `matrix2.txt` into another 2D array representing Matrix B of size ( m \times q ).

2. **Process Creation for Resultant Matrix Computation:**

   - The result of the matrix multiplication will be a matrix of size ( n \times q ). For each element ( C[i][j] ) of the result matrix, create a child process using `fork()`.
   - Each child process will be responsible for calculating the value of ( C[i][j] ).

3. **Nested Child Processes for Multiplication:**

   - Inside each child process (responsible for calculating ( C[i][j] )), create `m` nested child processes.
   - Each nested child process computes the product `A[i][k] * B[k][j]` for a specific `k` (where `0 <= k < m`) and returns the multiplication result using `exit()`.
   - The immediate parent process (responsible for computing ( C[i][j] )) will wait for all `m` nested child processes to complete, sum their results, and then exit using `exit()` to pass the computed sum back to the main parent process.

4. **Process Tree and Output**

   - Root Process(level 1) creates ( n \times q ) children (level 2) corresponding to each of the ( n \times q ) in the resultant matrix.
   - Each ( n \times q ) child on level 2 creates (m) children (level 3).
   - Each process on level 2 must print its PID and the resultant summation of the values returned by their (m) children
   - Each process on level 3 must print their PID and the resultant of their multiplication.

- The root process creates the final resultant matrix using the returned values by ( n \times q ) child processes and writes it to output.txt

5. **Modularisation**

- Create suitable functions to modularise the code where possible and submit it with a Makefile.
- Read Write functions are expected to be in a different file.
- If time permits modularise code of level 2 level 3 processes.

We will ensure no resultant no. is greater than 255. You can use the exit to return the results.

## Example of Matrix Multiplication Using Processes

### Matrix 1 (`matrix1.txt`, size 3x2):

```
1  2
3  4
5  6
```

### Matrix 2 (`matrix2.txt`, size 2x3):

```
7  8  9
10 11 12
```

### Resultant Matrix in output.txt (size 3x3):

```
27  30  33
61  68  75
95 106 117
```

### Terminal Output (There can be other possible outputs!)

```
Level 3 Process PID 63971: Computed 1 * 7 = 7
Level 3 Process PID 63972: Computed 1 * 8 = 8
Level 3 Process PID 63973: Computed 2 * 10 = 20
Level 3 Process PID 63975: Computed 2 * 11 = 22
Level 2 Process PID 63968: Sum for C[0][0] = 27
Level 3 Process PID 63976: Computed 1 * 9 = 9
Level 2 Process PID 63969: Sum for C[0][1] = 30
Level 3 Process PID 63978: Computed 2 * 12 = 24
Level 3 Process PID 63980: Computed 3 * 7 = 21
Level 2 Process PID 63970: Sum for C[0][2] = 33
Level 3 Process PID 63981: Computed 3 * 8 = 24
Level 3 Process PID 63982: Computed 4 * 10 = 40
Level 3 Process PID 63983: Computed 4 * 11 = 44
Level 2 Process PID 63974: Sum for C[1][0] = 61
Level 2 Process PID 63977: Sum for C[1][1] = 68
Level 3 Process PID 63985: Computed 3 * 9 = 27
Level 3 Process PID 63987: Computed 4 * 12 = 48
Level 3 Process PID 63989: Computed 5 * 7 = 35
Level 2 Process PID 63979: Sum for C[1][2] = 75
Level 3 Process PID 63991: Computed 5 * 9 = 45
Level 3 Process PID 63990: Computed 5 * 8 = 40
Level 3 Process PID 63993: Computed 6 * 11 = 66
Level 3 Process PID 63992: Computed 6 * 10 = 60
Level 3 Process PID 63994: Computed 6 * 12 = 72
Level 2 Process PID 63986: Sum for C[2][1] = 106
Level 2 Process PID 63984: Sum for C[2][0] = 95
Level 2 Process PID 63988: Sum for C[2][2] = 117
```

**Hints**

- Use waitpid()
- Use manpages for waitpid and read about WIFEXITED and WEXITSTATUS
- Use manpages for exit