Question 2

We have two dimensional array A, with indices (ip,jp) We want to compute an array B, were element B(i,j) is the sum of all elements in A for which ip is less than or equal to i and jp is less than or equal to j.

1. Write a serial program to do this.
2. Re-write program from step 1 as a parallel OpenMP program which uses tasks. Have one task to compute each entry of array B. Use normal loops to launch the tasks, with appropriate dependencies to ensure they execute in the correct order.
3. Having only one array entry per task will not be efficient. Adapt the program from step 2 so that it groups elements of the array into blocks, and then assign each block to one task.
4. Analyze and compare the performance of the three codes as the array size increases.

Logic

framework of program

key code segment

As explained in class, the tricky part is the following code: `for (ip = 0; ip < m; ip++) { for (jp = 0; jp < n; jp++) { if (jp == 0) b[ip * m + jp] = b[(ip - 1) * m + jp] + a[ip * m + jp]; else if (ip == 0) b[ip * m + jp] = b[ip * m + jp - 1] + a[ip * m + jp]; else b[ip * m + jp] = b[(ip - 1) * m + jp] + b[ip * m + jp - 1] + a[ip * m + jp]; } }` Element in matrix B can be calculated from "nearest box"

```
(base) ➜  question2 git:(main) ✗ ./sum_element_serial.x
Please enter array dimention(m * n):
3 3
Matrix A:
0       1       2
3       4       5
6       7       8
Matrix B:
0       1       3
3       8       16
9       24      48
```

How to parallelize the code

- task per entry
- task per block

Performance analysis

Failed to analyze in detail due to lack of time