COL730 Assignment 1: Successive Convolution with OpenMP

Problem statement:

Design and implement an OpenMP program that takes a 2-dimensional image and a filter of floating-point numbers and applies N successive convolutions using the same filter each time. The original image is used in the first iteration while the kth convolution is performed on output of the (k-1)th iteration for k=[2,N]. Each convolution is performed with a stride of 1 along both axes and the images are assumed to be padded with a value of 0.0 outside their valid range i.e. ([0..,X-1], [0..Y-1]), according to the following formula:

$$Img^{(k+1)}[i,j] = \sum_{u=0}^{U-1}\sum_{v=0}^{V-1} Img^{k}\left[i+u-\left\lfloor\frac{U}{2}\right\rfloor, j+v-\left\lfloor\frac{V}{2}\right\rfloor\right] \times h[u,v] \;\forall 0 \leqslant i < X, 0 \leqslant j < Y$$

Input format:

Input is read from a file named “input.txt”.

The first line of the input contains positive integer T representing the number of test cases.

The first line of each test case contains positive integers X Y U V N where (X, Y) and (U, V) are the dimensions of the image and filter respectively and N is the number of convolution iterations to be performed. The filter is guaranteed to have odd dimensions i.e., U, V are odd.

The following X lines contain Y floating-point numbers representing the image.

The following U lines contain V floating-point numbers representing the filter.

1 <= T <= 10

64 <= X, Y <= 1024

3 <= U, V <= 31

1 <= N <= 5

Output format:

Output is written to a file named “output.txt”.

For each test case, X lines follows each containing Y floating-point numbers representing the transformed image.

Submission Instruction:

Submit a single zip file with the following name [Your Entry Number].zip on moodle. The zip file should contain your source code along with a makefile that builds the required executable named “run”.

Check course webpage for last date of submission.

Note:

A virtual machine will be provided with 8 CPUs for testing purposes. The details will be announced soon.

All floating-point computations should be done using double precision floating-point arithmetic.

All floating-point inputs are formatted in scientific notation with 10 decimal digits of precision and the same format should be followed for all outputs.

The program is expected to scale with increasing number of CPUs available in the execution environment.