Mathematics u11601 (C Programming) Michaelmas 2021

November 17, 2021

Seventh assignment, due 12 noon, Wednesday 1/12/21

Plagiarism. The plagiarism policy is as always: you will not copy another student's assignment. If copying is detected, all students involved will lose marks, irrespective of who copied from whom.

Read this carefully. You should form the habit of reading specifications carefully, and following them.

The assignment is to write a C program, check that it works, and submit the C program.

Your program is to evaluate A^n efficiently, where A is a square matrix.

```
n ell m  (int: the exponent n and the dimensions of the array)
a_{00} a_{01} ... a_{ell-1,m-1} (the array: allow double precision)
```

- This is the last assignment with a fortnight allowed, to give extra time for debugging.
- You may assume that `ell == m`, the array is square.
- Your program should work with dynamically allocated 1- and 2-dimensional arrays, `quasi_2d` arrays (section 16). Working with pointers is very error-prone, and you may, for a reduced mark, work with standard 10×10 arrays. If you do so, your assignment will be marked out of 80 rather than 100.
- It is required that evaluation of A^n should follow the pattern of the Russian Peasant algorithm, as in Quiz 3.

Points to note.

The same remarks hold as in the previous assignments, especially that interactive input/output like

```
Please type in the next number:
```

should ***not*** occur.

Sample data (the results may be erroneous; if you suspect this, please email the instructor and/or tutors).

```
% # pow is a prototype implementation of assignment 7.
% cat mat1
2
2 2
1 2 3 4

% pow < mat1
Matrix A
    1.000    2.000
    3.000    4.000
raised to the power 2
matrix
    7.000   10.000
   15.000   22.000
% cat mat2
10
2 2
-8 6 -15 11
% pow < mat2
Matrix A
   -8.000    6.000
  -15.000   11.000
raised to the power 10
matrix
 -9206.000 6138.000
 -15345.000 10231.000
% pow < mat3
Matrix A
    1.000    1.000   -3.000
    1.000    3.000    1.000
    2.000    1.000   -2.000
raised to the power 5
matrix
   -7.000   53.000    9.000
  173.000  411.000  -19.000
   34.000  101.000  -22.000
```

```
% pow < mat4
Matrix A
    1.000    1.000    2.000    3.000
    2.000    3.000   -2.000   -2.000
   -2.000    2.000    1.000    1.000
   -3.000    1.000    2.000    3.000
raised to the power 9
matrix
 -252684.000 238961.000 -172070.000 -215916.000
 280156.000 -81677.000 123854.000 184124.000
 337913.000 -97958.000 -243543.000 -315262.000
 171276.000 -19075.000 -350630.000 -464492.000
% pow < mat5
Matrix A
    2.000    1.000    2.000    2.000    3.000
    3.000    1.000    3.000    2.000    2.000
    1.000    2.000   -2.000   -2.000    1.000
   -3.000    2.000    3.000   -3.000    3.000
    2.000    3.000   -1.000    2.000    1.000
raised to the power 4
matrix
  553.000  774.000  568.000  292.000  937.000
  564.000  806.000  675.000  322.000 1026.000
  500.000  374.000 -229.000  244.000   78.000
   52.000  532.000 -212.000 -303.000  318.000
  740.000  564.000  434.000  576.000  632.000
%
```