COSC2430 Hw1: Dense Matrix Multiplication

1. Introduction

You will create a C++ program to multiple two matrices, the operation is the same as linear algebra. You can use any algorithm to finish matrix multiplication, provided results are correct. It is preferred that your algorithm can multiple matrices efficiently, avoiding unnecessary products or data movement in the memory hierarchy.

Given two matrices A m x n and B j x k, you will compute result matrix C = AB.

2. Input and Output

The inputs are two text files, each should have one matrix. The output is a single text file, with one matrix.

File format: There will be ONE matrix row per line in the file for each input matrix. Each value should be a real number that may or may not have a decimal part (e.g. 1, 2.1, 3.1416). Values are separated by spaces. Besides digitals, other irrelevant characters will be given too, so your code should be able to report errors and output “error” (case sensitive).

3. Program and input specification

The main C++ program will become the executable to be tested by the TAs. The result matrix should be outputted to another text file, the name will be provided in the command line. Notice the input and output files are specified in the command line, not fixed as variables inside the C++ code. Notice also the format of the program call, to avoid Unix/Windows get confused, your code should be able to parse both kinds of command types.

The general call to the executable is as follows:

densemultiple "A=<filename>;B=<filename>;C=<filename>"

densemultiple A=<filename> B=<filename> C=<filename>

Above are the only calling method, no combination type of calling method will be given, such as densemultiple “A=<filename> B=<filename> C=<filename>” will not appears. The calling method will always be correct, will never appears the situation that missing one input file name or other similar errors.

Below are call examples.

densemultiple "A=a.txt;B=b.txt;C=c.txt"

densemultiple A=a.txt B=b.txt C=c.txt

The a.txt and b.txt are the file names of the input matrices, the c.txt is the output file name, which can be changed according to different test cases.

Assumptions: Matrices are given in dense form. Real numbers are separated by spaces. The output matrix should be written in dense form with a fixed two number of decimals.

Example 1 of input and result matrix
matrix11.txt (A=matrix11.txt)
1 1.a3
0 1

matrix21.txt (B=matrix21.txt)

```
1 0
0 1
```

output1.txt (C=output1.txt output file)

```
error
```

command line:

```
densemultiple A=matrix11.txt B=matrix21.txt C=output1.txt
```

or

```
densemultiple “A=matrix11.txt;B=matrix21.txt;C=output1.txt”
```

Example 2 of input and result matrix

matrix12.txt (A=matrix12.txt)

```
1 0 3 -1
2 1 0 2
```

matrix22.txt (B=matrix22.txt)

```
4 1 0
-1 1 3
2 0 1
1 3 4
```

output2.txt (C=output2.txt output file)

```
9.00 -2.00 -1.00
9.00 9.00 11.00
```

command line:

```
densemultiple A=matrix12.txt B=matrix22.txt C=output2.txt
```

or

```
densemultiple “A=matrix12.txt;B=matrix22.txt;C=output2.txt”
```

Example 3 of input and result matrix

matrix13.txt (A=matrix13.txt)

```
0.5 3.0 0.0 0.0
0.0 1.0 0.8 0.8
```

matrix23.txt (B=matrix23.txt)

```
1.0 6.0
0.0 2.0
2.0 0.0
```

output3.txt (C=output3.txt output file)
error

command line:
densemultiple A=matrix13.txt B=matrix23.txt C=output3.txt
or
densemultiple "A=matrix13.txt;B=matrix23.txt;C=output3.txt"

4. Requirements summary

1. It is encouraged you do not (not mandatory) use existing STL, vector classes since you will have to develop your own C++ classes and functions in future work.

   Your C++ code must be clear, indented and commented.
2. You must determine matrix size based on number of columns of each row of the input matrix. If deformed matrix appears, obviously, an error appears.
3. You can use static arrays of a maximum size=20. Your program will be tested with matrices up to 20 x 20. You can optionally use dynamically-sized 1-dimensional arrays (not dynamic 2D arrays since those require more careful manipulation) whose size is determined at run time.
4. Include comments in each source code file.
5. Your program will be tested with GNU C++. Therefore, you are encouraged to work only on Unix. You can use other C++ compilers, but the TAs cannot provide support or test your programs with other

compilers.

6. The output file must contain the result, in the same format (one row per line) as input or error message. Matrix values are written with 2 decimals separated by spaces. Do not use other separators or different number of decimals. Notice the number of decimals may vary in future homework.
7. Your program should be able to judge different errors, such as unmatched matrix size, illegal input characters, empty input files or other special situations. You should be able to assume different error types and design test cases by yourself to test your code.
8. Your program should write log messages to the standard output (cout, printf).

9. Turn in your homework

Homework 1 need to be turned in to our Linux server, follow the link here http://www2.cs.uh.edu/~rizk/homework.html.

Make sure to create a folder under your root directory, name it hw1 (name need to be lower case), only copy your code to this folder, no testcase or other files needed. If you use ArgumentManager.h, don’t forget to turn it in too.

ps. This document may have typos, if you think something illogical, please email TAs or Teacher for confirmation.