ACSE-5

Final Individual Coursework 2021

Answer the 2 questions in Part A (10+10 pts) and

1 out of the 2 questions in part B (10 pts) and

1 question in part C (15 pts)

1 out of the 2 questions in part D (15 pts)

1 question in part E (40 pts)

Total Marks: 100

Total time for activity: 3hrs

Section A: (Answer all questions) **(10 marks each)**

Question A.1

Consider the following integers: 9, 8, 7, 6, 5, 4, 3, 2, 1, 0. Use a container defined in the Standard Template Library to store these numbers. Write a single function that: removes all even numbers, sorts the numbers in the container into ascending order, and displays the numbers to screen.

Choose an appropriate container to perform these operations.

Question A.2

Consider the following definition of 'metallic means'.

The metallic means (also ratios or constants) of the successive natural numbers are the continued fractions:

$$n + \cfrac{1}{n + \cfrac{1}{n + \cfrac{1}{n + \cfrac{1}{n + \ddots}}}} = [n; n, n, n, n, \ldots] = \frac{n + \sqrt{n^2 + 4}}{2}.$$

Metallic means (Metallic ratios)

N	Ratio	Value	Class (Type)
0:	$\frac{0+\sqrt{4}}{2}$	1	
1:	$\frac{1+\sqrt{5}}{2}$	1.618033989[a]	Golden
2:	$\frac{2+\sqrt{8}}{2}$	2.414213562[b]	Silver
3:	$\frac{3+\sqrt{13}}{2}$	3.302775638[c]	Bronze
4:	$\frac{4+\sqrt{20}}{2}$	4.236067978[d]	
5:	$\frac{5+\sqrt{29}}{2}$	5.192582404[e]	
6:	$\frac{6+\sqrt{40}}{2}$	6.162277660[f]	
7:	$\frac{7+\sqrt{53}}{2}$	7.140054945[g]	
8:	$\frac{8+\sqrt{68}}{2}$	8.123105626[h]	
9:	$\frac{9+\sqrt{85}}{2}$	9.109772229[i]	
⋮			
n:	$\frac{n+\sqrt{4+n^2}}{2}$		

Create a function with the following declaration:

```
double computeMetallicMean(double n)
```

Write a programme that computes the values of the first ten metallic means as shown in the table.

Section B: (Answer 1 out of the 2 questions) **(10 marks)**

Question B.1

Write a programme that reads two 3x3 matrices from the terminal, in other words, you should write code to read matrices input from the standard input stream. Store these matrices using STL containers, and do not use pointers. Write a function that adds the two matrices without making a copy of the matrices. From your main function, print the entire operation to screen as follows:

```
MATRIX 1 = [1,2,3; 4.75,5,6; 1,-2,3]
MATRIX 2 = [1,0,0; 0,1,0; 0,0,1]
MATRIX 1 + MATRIX 2 = [2,2,3; 4.75,6,6; 1,-2,4]
```

Your programme should be able to add any two 3x3 matrices input by the user.

Question B.2

Read from a text file a sequence of (vegetable name, sow month, raised bed) tuples:

Leek	March	1
Carrots	April	1
Celery	April	1
Melon	March	1
Kohlrabi	February	2
Beans	March	2
Courgette	March	2
Nasturtiums	February	2
Cucumbers	April	3
Dill	April	3
Tomatoes	February	3
Nasturtiums	February	3

where the three entries on each line are separated by a space. Compute and print a list of vegetables that must be sown for each month, and report in which bed they must be sown. Your output to screen should be as follows:

```
Month: February
Vegetables to sow: Kohlrabi (Bed 2), Nasturtiums (Bed 2), Nasturtiums (Bed 3), Tomatoes
(Bed 3)

Month: March
Vegetables to sow: Leek (Bed 1), Melon (Bed 1), Beans (Bed 2), Courgette (Bed 2)

Month: April
Vegetables to sow: Carrots (Bed 1), Celery (Bed 1), Cucumbers (Bed 3), Dill (Bed 3)
```

Section C: (Answer 1 question) **(15 marks)**

Question C

Consider a tetrahedron defined by four nodes a, b, c, d.

The volume of a tetrahedron with vertices **a** = (a_x, a_y, a_z), **b** = (b_x, b_y, b_z), **c** = (c_x, c_y, c_z), and **d** = (d_x, d_y, d_z), can be computed by the following equation:

$$V = \frac{|(\mathbf{a} - \mathbf{d}) \cdot ((\mathbf{b} - \mathbf{d}) \times (\mathbf{c} - \mathbf{d}))|}{6}.$$

Part 1: Write a function that computes the volume of a tetrahedron using this equation. Write a programme (main function) that demonstrates the use of this volume function and outputs the results to screen.

Part 2: Create a Tetrahedron class with a constructor and a function that computes its volume. In your main function, create a loop that creates 10,000 Tetrahedron objects, computes their volume, averages them, and displays the total average to screen.

Clearly denote Part 1 and Part 2 in your code.

Section D: (Answer 1 out of the 2 questions) **(15 marks)**

For this section, please download the code on the ACSE-5 Teams files labelled "assessment_matrix.zip". This is a modified version of our Matrix and CSRMatrix classes from previous lectures. You are to modify these files and not another version of the Matrix or CSRMatrix classes.

Question D.1

You are to implement a method in the CSRMatrix class, called *dense2sparse*. This method creates a copy of the given dense Matrix<T> in a sparse format and hence returns a CSRmatrix<T> object (either as a return type or as an argument).

Please submit only your CSRMatrix.cpp file (and the .h if you have modified it) and a cpp file containing a main method that instantiates a Matrix<T> object and populates it. This dense matrix should have a number of non zeros smaller than the number of rows times the number of columns, i.e., a matrix that has some zeros in it. The main method should then call the dense2sparse method on that object and print the resulting sparse CSRMatrix<T> matrix.

Question D.2

You are to implement a method in the CSRMatrix<T> class, called *getDiagonal*. This method returns a shared pointer of type T[], which points to the first entry of an array stored in memory on the heap, which contains the diagonal of the matrix.

Please submit only your CSRMatrix.cpp file (and the .h if you have modified it) and a cpp file containing a main method that instantiates a CSRMatrix<T> object and populates it. This sparse matrix should have a number of non zeros smaller than the number of rows times the number of columns, i.e., a matrix that has some zeros in it. You should also not assume the CSRMatrix is square (i.e., your method must work if the number of rows is not equal to the number of columns). The main method should then call the getDiagonal method from the CSRMatrix and print the resulting array.

Section E: (Answer all questions) (40 marks)

For this section, please download the code on the ACSE-5 Teams files labeled "assessment_matrix.zip". This is a modified version of our Matrix and CSRMatrix classes from previous lectures. You are to modify these files and not another version of the Matrix or CSRMatrix classes (please modify new copies of these files for this question, don't simply add these methods to the files used in Question D).

Question E

Implement a copy constructor for the Matrix<T> class, that takes an existing Matrix<T> object as input and then instantiates and copies the values into the existing object. This copy constructor should not just produce a "shallow" copy of input (i.e., if the input is changed at some point, the values of the new object should not change).

In the Matrix<T> class, please write two versions of a matrix-vector product, one assuming that the matrix entries stored in the 1D flat array pointed at by the *values* pointer are in row major order, and another assuming they are in column major order. Call these two methods matVecMult_rowmajor and macVecMult_colmajor. They must both take two *double* pointers as arguments, representing the input and output vectors.

You must ensure your loop ordering respects the major orderings and uses contiguous memory accesses. Do not change the format of the *values* pointer (i.e., it must still be a flat 1D array).

Furthermore, write comments in your code identifying any potential level-1 BLAS calls that could replace for loops (e.g., daxpy or dot products). Please comment on what the arguments to those BLAS calls might be (i.e., identify the chunks of data that you could operate on all at once, like entire columns or entire rows). Also, note for both the row and column major matVecMults if this would change having to set the output to zero before you start your matVecMults.

Please submit only your Matrix.cpp file (and the .h if you have modified it) and a cpp file containing a main method that instantiates a Matrix<T> object and populates it. The main method should then copy the Matrix<T> using your new copy constructor, call the row major matVecMult on the first Matrix<T> object, and then call the column major matVecMult on the second Matrix<T> object. Print both outputs from your matVecMults.

Hint: the row and column major matVecMults should return **different** results if called as above; if we name the original Matrix<T> **A**, then the row major will return: output = **A** * input, the column major will return the equivalent of: output = transpose(**A)** * input).