ACSE-5

Final Individual Coursework 2020

Answer 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

Write a **function** that takes in an integer and then reverses the order of the digits in the integer. For instance, changes “12345678” to “87654321”. The main section of the program should allow the user to enter an integer, it should then call the function to reverse the integer’s order and display the resultant integer.

Question A.2

Write a function that concatenates two strings. Use **a pointer** as the argument type. Write another swap function using **a reference** as the argument type.

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

Question B.1

Two quantities are in the **golden ratio** if their ratio is the same as the ratio of their sum to the larger of the two quantities. It follows that a and b are in the golden ratio if:

$$\frac{a+b}{a}=\frac{a}{b}=1.618$$

Write a function that reads a single number, z, from the screen and computes two numbers, x and y, such that their sum is the original input number, $x+y=z$, and they are in the golden ratio, $x/y=1.618$. The function

Question B.2

Read from a text file a sequence of (name, value, Boolean) tuples, for example:

John	101.7	1
Jane	21	1
Pippa	58.78	0
John	50	0
Lawrence	103.98	1
Ellie-May	67.83	1

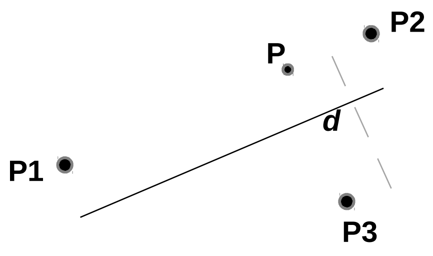
where the name is a single whitespace-separated word (John), the value is an integer or a floating-point value (101.7), and the Boolean is a 1 or a 0 value. Compute and print the sum and mean for all names for which the Boolean value is true (the mean would be = (101.7+21+103.98+67.83)/4), and the sum and mean for all names (the mean would be = (101.7+21+58.78+50+103.98+67.83)/6).

Section C: (Answer all questions) **(15 marks)**

Question C

The solution to finding the shortest distance from a point to a line segment is defined by the equation of a line defined through two points **P1** (x1, y1) and **P2** (x2, y2)

$$P = P1 + u(P2 - P1)$$



where

$$u = \frac{(x3 - x1)(x2 - x1) + (y3 - y1)(y2 - y1)}{\|P2 - P1\|^2}$$

And so, P = (x, y) can be defined as:

$$x = x1 + u(x2 - x1) y = y1 + u(y2 - y1)$$

And, the distance between the point **P3** and the line is the distance between **P** and **P3**. Write a function that computes the distance between a segment and a point. Write a programme (main function) that demonstrates the use of this distance function, and outputs the results to screen.

Note: ensure that your function will not crash if the two points are coincident (i.e., have the same x and y coordinates).

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

For this section, please download the code on the ACSE-5 github labelled "assessment_matrix.zip". This is a modified version of our Matrix and CSRMatrix classes from previous lectures.

Question D.1

You are to implement a method in the CSRMatrix class, called *sparse2dense*. This method creates a copy of the given sparse CSRMatrix<T> in a dense format and hence returns a matrix<T> object.

Please submit only your CSRMatrix.cpp file 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. The main method should then call the sparse2dense method on that object and print the resulting dense Matrix<T> matrix.

Question D.2

You are to implement a method in the CSRMatrix class, called *computeTrace*. This method computes the trace of the sparse matrix, which is the sum of all diagonal entries. This method should return only a double value containing the trace of the matrix.

Please submit only your CSRMatrix.cpp file 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. *Also, some of those zeros* ***should*** *be on the diagonal*. The main method should then call the computeTrace method on that object and print the resulting trace value.

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

For this section, please download the code on the ACSE-5 github labelled "assessment_matrix.zip". This is a modified version of our Matrix and CSRMatrix classes from previous lectures.

Question E

Implement a copy constructor for the CSRMatrix class, that takes an existing CSRMatrix<T> object as input and then instantiates and copies the values into the existing object. This copy constructor should not be 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).

As part of your work on linear solvers, you have decided to implement an incomplete LU factorisation, which computes matrices such that for a given matrix **A ≈ LU**, where L and U are lower and upper triangular matrices, respectively. Writing an ilu(0) method can be difficult however, so you decide to just implement part of an ilu(0) method, given by the following pseudo-code:

If **A** is a matrix of size nxn:

```
for i = 2 to n
    for k = 1 to i-1
        A[i][k] = A[i][k] / A[k][k]
```

Implement this algorithm for the CSRMatrix class, by creating a method which takes no arguments and modifies the values of the matrix **A** "in-place", i.e., by overwriting the values of **A**. Do not at any point convert **A** to a dense matrix, or use an amount of memory equivalent to **A** stored densely, you must operate on the sparse matrix.

Please submit only your CSRMatrix.cpp file 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. *Also, **none** of those zeros should be on the diagonal*. The main method should then copy the CSRMatrix using your new copy constructor, and then compute an ilu(0) on the new copy. Print both the original CSRMatrix and the newly decomposed matrix.