CSE108 – Computer Programming Lab.

Lab 8

Due at 10am.

Hand in: A student with number 20180000001 should hand in a zip file named 20180000001.zip for this lab.

PART 1. [30]

void print_matrix(matrix initial_matrix)

Create a function that accepts a matrix structure type as an argument. No other data types or variables should be present in the matrix structure type, only a 2D double array representing a real 3x3 matrix and a double variable indicating its determinant. This function must be used in main(...) and should print the matrix in a pleasing manner (centered entries with no more than four digits following the decimal point, as shown in the example below).

```
1.0000    0.9134    0.2785
0.9058    0.6324    0.5469
0.1270    0.0975    0.9575
```

void inverse_matrix(matrix* initial_matrix, matrix* inverted_matrix)

Implement this function, which should be used after print_matrix(...) in main(...). The function must first call the subsequent function and then store its determinant in the relevant variable inside the type of the matrix structure;

void determinant_of_matrix(matrix* initial_matrix)

Then, if the 3x3 matrix is invertible, the inverse_matrix function should locate the matrix's inverse as well as its determinant (keep in mind that determining the inverse requires calculating the determinant). The specified variables in inverted_matrix are where the calculated values should be kept. Last but not least, use print_matrix in main(...) to print the inverted matrix if it is invertible. Whenever an invertible matrix is encountered, simply print a suitable error message in main(...). Any additional helper functions you create on your own are usable. You will require the following formulas:

Determinant of A matrix $M=\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$ given by $|M| = a\begin{vmatrix} e & f \\ h & i \end{vmatrix} - b\begin{vmatrix} d & f \\ g & i \end{vmatrix} + c\begin{vmatrix} d & e \\ g & h \end{vmatrix}$.

The inverse of a matrix is given by $M^{-1} = \frac{1}{|M|}\begin{bmatrix} \begin{vmatrix} e & f \\ h & i \end{vmatrix} & -\begin{vmatrix} d & f \\ g & i \end{vmatrix} & \begin{vmatrix} d & e \\ g & h \end{vmatrix} \\ -\begin{vmatrix} b & c \\ h & i \end{vmatrix} & \begin{vmatrix} a & c \\ g & i \end{vmatrix} & \begin{vmatrix} a & b \\ g & h \end{vmatrix} \\ \begin{vmatrix} b & c \\ e & f \end{vmatrix} & -\begin{vmatrix} d & e \\ g & h \end{vmatrix} & \begin{vmatrix} a & b \\ d & e \end{vmatrix} \end{bmatrix}^T$ where T indicates transpose. Obviously, the inverse is not defined if the matrix has zero determinant.

2.PART [40]

double find_orthogonal(vector vec_1, vector vec_2, vector* output_vec)

Create a function that accepts a type argument of the vector structure. Only the three double-digit x, y, and z dimensions of a 3D vector should be included in the vector. The angle in degrees between the input vectors vec_1 and vec_2 should be calculated and returned by this function. The vector cross product should locate the vector orthogonal to the input two vectors and return it as the output argument. The found angle and vector ought to be shown in main(...). Functions from the math.h library can be used.

For two given vectors $\vec{A} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}$ and $\vec{B} = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}$, the angle between is given by $\theta = \cos^{-1} \frac{\vec{A}.\vec{B}}{\|\vec{A}\|\|\vec{B}\|}$

(where $\|\vec{A}\| = \sqrt{a_x^2 + a_y^2 + a_z^2}$ and $\vec{A}.\vec{B} = a_x b_x + a_y b_y + a_z b_z$) and the cross product by $\vec{A}\ x\ \vec{B} = \begin{bmatrix} \begin{vmatrix} a_y & a_z \\ b_y & b_z \end{vmatrix} & -\begin{vmatrix} a_x & a_z \\ b_x & b_z \end{vmatrix} & \begin{vmatrix} a_x & a_y \\ b_x & b_y \end{vmatrix} \end{bmatrix}^T$

3.PART [40]

polynomial get_integral(third_order_polynomial p1, third_order_polynomial p2, int a, int b)

Write a function that takes **third_order_polynomial** structure type as arguments. **third_order_polynomial** structure type should contain neither variable nor data type other than double variables representing coefficients of the third-degree polynomial (for $ax^3 \neq 0$). In this task, the program should ask for the user to input two third-degree polynomials and interval values. Between specified intervals[a, b], by calculating the integral of multiplication of these polynomials, it should return a new polynomial structure type containing coefficients (including the constant) and its value between [a, b] of the integrated polynomial. polynomial structure type should contain neither variable nor data type other than double variables representing coefficients of the integrated polynomial and a char variable indicating constant. In addition to this, it should also contain a double variable representing its value between [a, b]. Found coefficients and the value between [a, b] should be printed inside main(...).

General Rules:

1. You will have two hours to provide a solution to the given problem set.
2. You will be able to hand in your solutions via Teams in the next two hours. The submission will be closed exactly at 10am.
3. There will be an interview session immediately after the submission deadline. Starting at 10am, you will be randomly invited to attend a meeting by a TA to demonstrate your solution and answer any questions asked by the TA.
4. You must be available until 1pm to respond to the demo invitation whenever you receive it. You will have 3 minutes after you are called via Teams. If you do not answer/appear in 3 minutes, you will miss you interview.

5. If you miss your interview or are unable to give satisfactory answers to the questions, you will receive a zero for that lab even if you have submitted your solution.
6. If you have not submitted a solution in time, you will not be invited for the interview and receive zero for that lab.
7. Due to time constraints, some students may not be invited to an interview. In that case, their solutions will be graded offline.
8. Unless you aren't declared for a specific prototype, you may use arbitrary but proper function and variable names that evoke its functionality.
9. The solution must be developed on given version of OS and must be compiled with GCC compiler, any problem which rises due to using another OS or compiler won't be tolerated.
10. Note that if any part of your program is not working as expected, then you can get zero from the related part, even it is working partially.
11. Zip your solution file before uploading it to MS Teams. The zip file must contain the C file with your solution and screenshots of the valid outputs of the program.