PROG20799 Data Structures

Assignment #2

(Due Date: See SLATE)

Prof. Georg Feil / Winter 2020

Vector Math (12 marks / 5%)

This assignment will help you get practice with structures, pointers, and passing structures to functions. You'll implement a program that performs math on 3D vectors.

Download and examine the header file `vector_math.h` from the dropbox. You must write a corresponding `vector_math.c` source file which implements all the functions whose prototypes are given in the header file (don't forget to include the header file from your .c file).

For this assignment you should do the following:

1. You have to add a few definitions to the header file `vector_math.h`. Specifically, add a typedef which makes `Point` an alias for `Vec3d`, and add the declaration of the `Line` structure (using `Point`). Do not make any other changes. See the header file and carefully read the comments for details.

2. Implement functions in your .c file for all prototypes given in the header file.
   Note: To add two vectors, add the x coordinates together and store the result in the destination vector's x coordinate. Repeat for the y and z coordinates.
   To multiply a vector by a scalar 'a', multiply each vector coordinate by 'a'.
   When printing a vector, I recommend you use %g, not %f.
   The line functions are more tricky, so get all the code for vectors working first!

3. Write a main function which does the following, in the order given below:
   a) Create four `Vec3d` variables (don't use malloc). You may optionally use an array.
   b) Set the first vector to (-2, 1, 11) by calling the `vec3dSet` function. Remember to use '&' to pass variables as pointers when needed!
   c) Set the second vector to (3.5, -7, 0.3) using the `vec3dSet` function.
   d) Add the first vector to the second vector, and store the result in the third vector using the `vec3dAdd` function.
   e) Multiply the third vector by the scalar 0.5 using the appropriate function and store the result in the fourth vector.
   f) Print all four vectors using the `vec3dPrint` function.
   g) Create a `Line` (don't use malloc).

h) Use the first and fourth vector to set the starting and ending points of the line by calling the `lineSet` function.
i) Print the line using the appropriate function.
j) *Dynamic memory allocation:* Create two `Vec3d` variables that are allocated dynamically using malloc. Hint: The data type should be "pointer to `Vec3d`".
k) Assign the value of the first vector created above [steps a, b] to the first new malloc'd vector. Must be done using one statement. Use the assignment operator '=' or the vec3dSet function.
l) Multiply the first malloc'd vector by the scalar -1.7 and store the result in the second malloc'd vector using `vec3dScalarMul`.
m) Print both malloc'd vectors using `vec3dPrint`. Don't forget to free the dynamically allocated vectors before your program quits!

The output of your program should look like this:

```
(-2, 1, 11)
(3.5, -7, 0.3)
(1.5, -6, 11.3)
(0.75, -3, 5.65)
(-2, 1, 11) - (0.75, -3, 5.65)
(-2, 1, 11)
(3.4, -1.7, -18.7)
```

Assignment Submission

Your assignment submission must follow all the requirements outlined below.

Do your work individually. This is not a group assignment. Sheridan's rules on academic integrity must be followed. If you have questions about this assignment please ask me in class, or come and see me!

Submit your assignment using SLATE (click on the Dropbox tab, then click on the correct assignment link). Attach all source files (.c, .h) to your submission. Do not use ZIP or RAR.

Your assignment must be submitted before the due date/time. Any assignments submitted after this time are considered late. See our class plan on SLATE for my rules governing late assignments.

Evaluation

Your submission will be evaluated based on the following criteria:

Functionality: Program functions according to requirements and is structured as specified in the assignment description. A program that does not compile will automatically receive a grade of ***zero***. If any warnings are displayed using -Wall and other required compiler options given in the week 1 slides then one mark will be deducted. If necessary comment out code that gives syntax errors, I may give you part marks.

Programming Style: Proper indentation, spacing, and use of comments. All identifiers are descriptive (avoid one-letter variable names). Variables are defined with appropriate data types and initialized correctly. Variables use appropriate scope (local/module/global). For additional programming style rules see our coding standards on SLATE.

Efficient Code: Program doesn't use too much repetitive code (e.g. uses functions instead of copy/pasting code). Program uses global variables where and only when necessary. Program doesn't define variables that are never used, nor does it use too many variables for unnecessary tasks. Program logic is written concisely and is not cluttered with unnecessary code.

External Documentation (if any): Good organization and clarity of written communication.

Other: All instructions regarding assignment submission and program specifications have been followed. Submission was completed as requested in a timely fashion. Techniques discussed in class have been used as appropriate.