CPRE 185 – Introduction to Computer Engineering and Problem Solving I
Spring 2018, Iowa State University
100 Points

HOMEWORK 10 & 11 – Structs, Pointers

Submission Instructions

1. **Submission** – Submit through Blackboard by **Apr 24, 2018 3.00 PM**
   ONLY 3 SUBMISSION ATTEMPTS ALLOWED IN CANVAS
   ONLY LAST SUBMISSION WILL BE GRADED

2. **Homework submission – Theory questions must be neatly typed and submitted through Canvas as WORD or PDF document only.**

3. **Source code submission – Source code must be uploaded as C file with .c extension for coding exercises. Append the homework number and problem number to the source code file name. For example, *cpre185_hwk1_problem2.c* .**

4. **DO NOT paste source code in WORD/PDF file. Zero credit will be given for source code uploaded as WORD/PDF file.**

5. **Concise and meaningful comments must be provided for the instructor/grader to understand your source code. Failure to include adequate comments in your code will result in deduction of 50% of the maximum credit for the coding exercise.**

6. **Include the following at the TOP of your source code file (using C style comment syntax). Failure to include the below in your code will result in deduction of 50% of the maximum credit for the coding exercise.**

```
/*********************************************/
/*** CPRE 185 – Homework 'Put homework number HERE' ***/
/*** *****************************************/
/*** Author – 'Your Full Name'  ****/
/*** Lab section – 'Your Lab section'  ****/
/*** Date started – 'Date started coding' ****/
/*** Comments – 'general comments about this code' *****/
/*********************************************/
```

7. **Zybook exercises – Complete Zybook activities in the E-book itself. Do not upload Zybook exercises on Blackboard. Zybook exercise are for your practice only. These exercises will NOT be graded.**

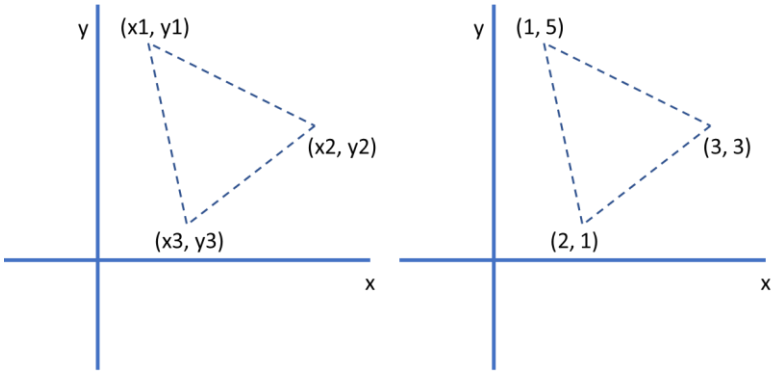
8. **NOTE: Use meaningful variable names. Use parenthesis to group terms in expressions for better readability.**

9. **Provide meaningful information to user when expecting scanf inputs. For example, '*Enter your height:*' before expecting an input through scanf.**

10. **Code MUST compile and work completely for receiving FULL points. Partial points may be awarded at the discretion of the grader.**

1. Write C code for the following. Prompt for inputs as needed by your code. Assume points are represented as ***float*** values.

   a. A triangle is represented by the cartesian co-ordinates of three points shown in the below figure. [10]

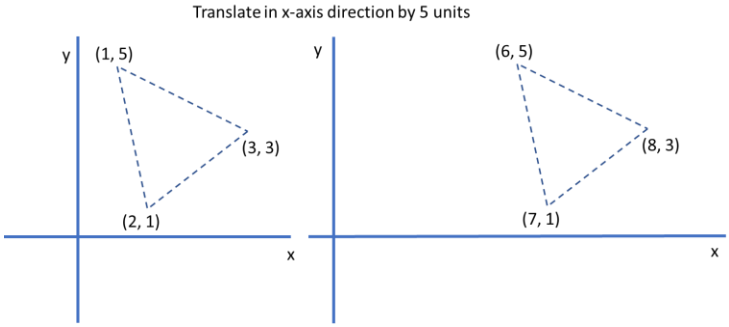


   Define a ***struct Point*** for representing a point (x,y) in the **x-y** 2D space. Use the Point struct to define another **struct Triangle** to represent a triangle (represented by 3 points) in your C code. Hence, your triangle struct will have 3 Point struct as members, one for each point.

   b. Write a function ***read_triangle*** which reads three points describing a triangle into a struct variable and returns the struct. [10]

   c. Write a function ***print_triangle*** which prints the contents of a struct variable describing a triangle. You must use a function ***print_point*** to print the three point structs which are part of the triangle struct. [20]

   d. Write a function ***translate_triangle_xdir*** which translates the triangle along the x-axis and returns the translated triangle struct. X-Translation is moving the triangle along the x-axis by some distance, ***d*** units. This is the same as adding ***d units*** to the x-coordinate of each point in the triangle. [20]



   e. Write a function ***translate_triangle_ydir*** which translates the triangle along the y-axis and returns the translated triangle struct. Y-Translation is moving the triangle along the y-axis by some distance, ***d*** units. This is the same as adding ***d units*** to the y-coordinate of each point in the triangle. [10]

f. Use the functions in steps **d** and **e**, to write a function ***translate_triangle_xydir***, which translates a triangle in both x and y directions, and returns the translated triangle struct. [10]

g. Write a function ***compare_perimeter_triangle*** which takes two triangles as parameters and compares the perimeter of both triangles. Return 1 if the perimeter is same with some tolerance, 0 otherwise. [20]

Perimeter of triangle = Sum of the length of all three sides

See below for computing the distance between two points in cartesian space.
http://www.purplemath.com/modules/distform.htm

2. Write C code for the following. (See section 8.6 in Zybooks) MUST use pointers in this exercise.

a. In C, char data type requires 1 byte of memory for each character. Scan in an integer, ***nchars***, in your code.

b. Allocate memory for a character array of size ***nchars***. User will input ***nchars*** during code execution.

c. Prompt for characters to initialize the dynamically created array in Part b. using a loop.

d. Print the contents of the character array using a loop.

3. Readings in Zybook on Pointers – Chapter 8 – Section 8.1 to 8.6