Lab 9: Structures...

**Due date:** Monday, December 1, 11:59pm.

# Lab Assignment

## Assignment Preparation

**Lab type.** This is an individual lab. During the Tuesday, November 25 lab period, students are allowed to consult their peers. However, each student has to develop his/her own code.

**Purpose.** This lab allows you to get acquainted with C's `structs`.

**Programming Style.** All submitted C programs must adhere to the programming style described in detail at

http://users.csc.calpoly.edu/∼cstaley/General/CStyle.htm

When graded, the programs will be checked for style. Any stylistic violations are subject to a 10% penalty. Significant stylistic violations, epsecially those that make grading harder, may yield stricter penalties. Also note the the Lab 2 requirement for the content of the header comment in each file you submit applies **to each assignment** (lab, programming assignment, homework) in this course.

**Testing and Submissions.** Any submission that does not compile using the

```
gcc -ansi -Wall -Werror -lm
```

compiler settings will receive an automatic score of 0.

# Assignment

You will develop three programs that use `structs` to store data.

## Program 1: auction.c

Write a program that accepts information about a number of items sold at an auction, stores this information internally as `structs`, and outputs some information about the data acquired.

**Program input.** The input to the program will consist of description of three auction items. Each description will come in a single line (i.e., the "
n" character will appear only at the end of each auction item description in the input). For each item, the following information will be provided;

1. Type of the item. E.g., `"painting"`, or `"gift_basket"`. The type of the item is a string of at most 20 characters containing no spaces.
2. Starting price. A positive integer number.
3. Number of bids. A positive integer number.
4. Type of auction. A string. Acceptable values are either `"live"` or `"silent"`.
5. Winner of the auction. A string (e.g., `"John_Brown"`). The string will contain at most 30 characters and will contain no spaces.

**Example.** The following input:

```
painting 100 5 live Joe_Smith
gift_basket 50 3 silent Anna_Brown
concert_tickets 40 12 live Janet_McDermott
```

specifies three auction items. The first item, a painting with a starting price of $100, and the third item, a pair of concert tickets with a starting price of $40 (for both) were offered at a live auction and attracted five and 12 bids respectively. The second item, a gift basket, was offered at a silent auction with a $50 starting price. It attracted three bids. The painting was won by Joe Smith, the gift basket — by Anna Brown and the concert tickets —- by Janet McDermott.

**Data structures.** You shall define a C structure type to store information about individual auction items. Your structure shall contain all information about an auction item.

**Program flow.** Your program shall read from standard input the descriptions of the three auction items. Each description shall be stored in a `struct` variable. Your program shall output the following information:

- The item that fetched the largest number of bids. Report the type of the item, the starting price, the number of bids, the sale price and the winner of the bidding for the item.

  The final price of an item is computed as follows: for live auctions, each bid after the first increases the price of the item by $5. For silent auctions, each bid after the first increases the price of the item by $2.

  **Example.** For the input above, the output will be:

```
Most popular: conert_tickets   starting price: $40  ending price: $95
   Attracted: 12 bids.  Winner: Janet_McDermott
```

- The item with the highest final sale price. Report the item type, type of the auction, the starting and ending prices, and the winner of the auction.

  **Example.** For the input above, the output will be:

```
Most expensive: painting
    Offered at: live auction
Starting price: $100     ending price: $120
        Winner: Joe_Smith
```

- Total number of items offered at each type of auction, and total number of bids in each type of auction.

  **Example.** For the input above, the output will be:

```
  Live auction: 2 items, 17 bids
Silent auction: 1 item, 3 bids
```

  **Note:** Make certain, you use correct forms of the words "item" and "bid" in the output. If the number of items is equal to one, the output shall be "1 item", in all other cases, use "items". Similarly, if there was only one bid registered at the auction type, the output shall contain "1 bid", while in all other cases it shall be "bids".

Include an empty line between each of the three output portions.

**Example.** The full output of your program, given the input above will be:

```
Most popular: conert_tickets   starting price: $40  ending price: $95
   Attracted: 12 bids.  Winner: Janet_McDermott

Most expensive: painting
    Offered at: live auction
Starting price: $100     ending price: $120
        Winner: Joe_Smith

Live auction: 2 items, 17 bids
Silent auction: 1 item, 3 bids
```

**Error checking.** Your program shall guard all varaibles. If any of the inputs contains an incorrect value, the program shall terminate without any attempts at further activity. An apprornpiate diagnostic message, specifying the name of the parameter and the fact that its value is incorrect shall be printed.

For example, if the first line of the input is

```
blanket -20 1 live Bob
```

your program shall output

```
Incorrect starting price: -20
```

## Program 2: triangle.c

You shall write a program that inputs the coordinates of three points in a three-dimensional space, and outputs the perimeter of the 3D triangle with these points as vertices. Your program shall include the following:

- Use of C's `struct` variables to store each of the three points. You shall define an appropriate `struct` data type.
- A function that takes as input two `struct` variables representing two points in 3D space and returns back the distance between the points.

**Note**: all input parameters shall be treated as `float` values. The function computing the distance between two points shall return a `float` value as well.

**Example.** Given the following input:

```
0.0 0.0 1.0
0.0 1.0 0.0
1.0 0.0 0.0
```

your program will output:

```
4.242640687
```

(note, each side of this triangle has length $\sqrt{2}$; $3 \cdot \sqrt{2}$ is approximately 4.24260687. Your answer may be different by a fraction).

## Program 3: grading.c

You will write a program that takes as input the list of students in a class, together with their grades for various coursework, and outputs for each student, his/her final score in the course.

**Input.** The input to the program is a list of students. The entry for each student will be on a single line, entries for different students will be spearated by a line break. For each student the following information is given:

1. Student's last name, followed by their first name. Both names are strings; the maximum size of both first and last name is 20 characters.
2. Five floating point numbers indicating, respectively, student performance on two homeworks (HW1 and HW2), course project, midterm and final exam. All scores are out of 100 points.

The end of input will be marked by a line consisting of seven zeroes.

**Example.** The following input:

```
CAR MAUDE 100.0 70.5 88.5 65.5 77.5
KRISTENSEN STORMY 68.0 59.0 69.0 82.5 73.9
VANDERWOUDE SHERWOOD 40.0 50.0 65.0 70.0 68.0
GELL TAMI 84.0 95.0 87.9 100.0 96.5
MADLOCK RAY 45.0 67.5 100.0 100.0 90.0
0 0 0 0 0 0 0
```

describes the performance of five stuents in the course.

**Output.** Your program shall output the name of each student, their overall score in the course (out of 100) and the letter grade the student received.

**Computing the course score.** Your program shall use a function that takes as input necessary parameters and returns a `float` number indicating the course score of the given student. The course score is determined using the following weights for individual course work:

| Coursework | % of final grade |
|---|---|
| Homework 1 | 10% |
| Homework 2 | 15% |
| Project | 25% |
| Midterm | 20% |
| Final exam | 30% |

**Computing the letter grade.** Your program shall use a function that takes as input necessary parameter(s) and returns a `char` letter grade.

The letter grade is determined as follows:

| Score range | Letter grade |
|---|---|
| $[85, 100]$ | `A` |
| $[75, 85)$ | `B` |
| $[60, 75)$ | `C` |
| $[50, 60)$ | `D` |
| $[0, 50)$ | `F` |

**Data structures.** You shall define a `struct` type to store information about individual students. You shall use an array of such `structs` to store all input information. The will be no more than 100 student descriptions in the input, although there may be fewer.

**Error checking.** Each score must be checked. The range of valid values for each score is [0,100]. If an invalid score is encountered, the program shall terminate without any further output.

**Example.** For the input above, the program shall produce the following output.

```
CAR, MAUDE: 78.00, B
KRISTENSEN, STORMY: 69.00, C
VANDERWOUDE, SHERWOOD: 61.00, C
GELL, TAMI: 91.00, A
MADLOCK, RAY: 86.00, A
```

(Note: use `"%2.2f"` in the format string of the `printf` statement to get the score printed with two digits after the decimal point.)

# Submission procedure.

You need to submit three files: `auction.c`, `triangle.c` and `grading.c`.

You will be using `handin` program to submit your work. The procedure is as follows:

- `ssh` to `vogon` (`vogon.csc.calpoly.edu`).
- Students from **Section 009** shall execute the following submission command:

  ```
  > handin dekhtyar-grader lab09-09  <your files go here>
  ```

- Students from **Section 011** shall execute the command:

  ```
  > handin dekhtyar-grader lab09-11 <your files go here>
  ```

`handin` is set to stop accepting submissions 24 hours after the due time.