CS 211 Lab 3 – Summer 2023

Logistics

For this lab you will edit the file lab3.c to work with structs, arrays, and pointers. Submit your edited program to Gradescope.

Important Notice: Your programs will be tested on *bertvm* servers. If your program does not compile or run on *bertvm* you will lose points. Make sure all lab assignments are completed (or at least tested) on *bertvm* before submission.

Overview

In this lab, you will explore structures, arrays, and pointers in the C language. Follow the instructions and complete the given tasks in the lab3.c file.

By the time you have completed this lab, you should be able to:

- Declare and use C structs
- Access structure members using dot (.) and arrow (−>) operators.
- Pass structs to and from functions and use arrays of structures

Instruction

Lab3.c contains a structure that represents a data sample as might be collected by a scientific drone, lunar rover, or autonomous underwater vehicle, AUV:

```
struct SAMPLE {
        int ID;                 // Unique serial number for each Sample.
        double x, y;            // Location where sample was collected.
        double data;            // Data value at the sample point.
};
```

Task 1: Use typedef to convert the structure declaration to a new type, Sample. For the rest of this program we will declare instances of this struct using Sample.

Task 2: Edit the two print functions, printSampleByValue( ) and printSampleByAddress( ) to print data values from the variable passed into them, instead of the constants that are there.

Next, in the main function we declare an instance of Sample struct, s1.

Task 3: Initialize s1 using initSample( ), and print it out using printSampleByValue( ) .

Compile your program and run it at this point. Fix any errors before proceeding

Task 4: Declare a pointer of type Sample * called ptr1.

Task 5: Allocate memory for the Sample pointed to by ptr1, initialize it by calling initSample( ), and print it out using printSampleByAddress( ).

Compile your program and run it at this point. Fix any errors before proceeding

Create two more instances of Sample struct declared as s2 and s3, and initialize s2 using initSamples( ).

Task 6: Edit averageSamples( ) to create a new Sample and set its values equal to the average of the two Samples passed in as arguments. (Call initSample( ) on the new Sample first, to set the ID value. Then calculate the x, y, and data values as the averages of the corresponding values from the two input Samples.)

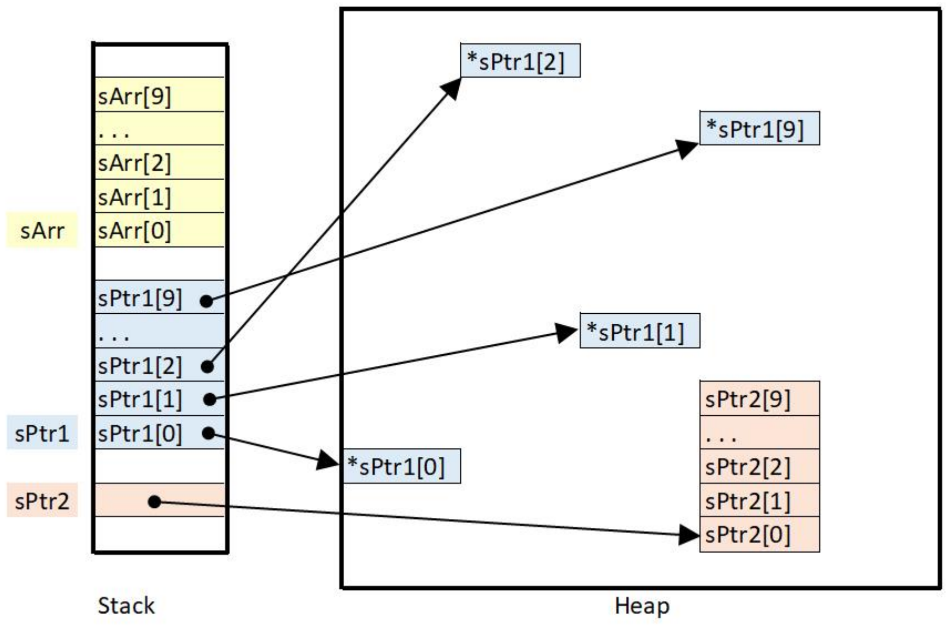
Task 7: Use averageSamples( ) to make s3 the average of s1 and s2. Print all 3 samples with suitable print statements, and check that the values in s3 are indeed the average of s1 and s2.

Compile your program and run it at this point. Fix any errors before proceeding

Next, we are going to look into the concept of arrays of structures. The following three variable declarations all could be used to allocate some form of an array with 10 Samples. Read the descriptions of what is being allocated by each statement:

```
Sample     sArr[ SIZE ];     // Array of Samples on the stack.
Sample * sPtr1[ SIZE ];      // Array of pointers on stack to Samples in the heap.
Sample * sPtr2;              // Array of Samples on the heap
```

If properly created/allocated, these arrays in memory will look something like the following:



Task 8: For array sArr, uncomment and fix the print statement.

Task 9: For array sPtr1, properly allocate and initialize it and display the results using one of the print functions.

Task 10: Repeat the same step for sPtr2, allocate, initialize and display the results.

Submission

Once you have completed all the tasks, submit your complete lab3.c on Gradescope.