CSCD 340
Lab 1

You answers will be in the form of a PDF/C Code. You will submit a zip file named your last name first letter of your first name lab1.zip. (Example steinerslab1.zip)

NOTE 1: For this lab the answers for problem 1 through problem 4 will be placed in a single PDF file named lab1answers.pdf. You must label the problem and the question then your answer. Failure to do so will result in 0 points per problem.

1. Provided is a C file named cscd340_s18_lab1prob2.c Within the C file are printf statements. Use the starting address for a, b, c, and i starting addresses below.

   For each printf statement explain what you believe will be displayed. I want an educated guess. Your output for this will be similar to the following:

   - Your output for this will be similar to the following:
     1: a = 0022FF00, b = 0022FEFC, c = 0022FEF8, i = 0022FEF4

     2: b = 002D0F98, c = FFFFFFFE
   - Save these guess as a question #1 in your PDF.

2. After you have made an educated guess for question 2, compile and execute the code. Copy your output and either verify you were correct or explain where you went wrong. Your output for this will be similar to the following:

   Given the starting addresses of:

   a = 0x7fffd191bc10, b = 0x7fffd191bc08, c = 0x7fffd191bc00, i = 0x7fffd191bbfc

   My program produced: a=0x7fffd0517680 b=0x????????

   Although the addresses are different I was correct in how many bytes were allocated between a, b, c and i.

   NOTE: For #6 just give me your best guess and why you think that value is correct.

3. It is important that you have a fundamental understanding of pointers and pointer arithmetic, especially the differences between the HEAP and the stack. To illustrate that understanding answer the following

   a) Does the stack grow up or down? How do you know? Justify your answer.
   b) What version of GCC are you using?
   c) What is odd about how memory is arranged compared to the declarations, especially the declarations in main?
   d) Does the array declaration, in main, leak memory? Why are why not? (Here I am looking for where array resides in memory)
   e) Does the calloc, in main, leak memory? Why are why not? (Here I am looking for where the variable resides in memory) If yes how many bytes?
   f) Does the malloc, in foo, leak memory? Why are why not? (Here I am looking for where the variable resides in memory) If yes how many bytes?
   g) Does the calloc, in foo, leak memory? Why are why not? (Here I am looking for where the variable resides in memory) If yes how many bytes?
   h) Run the program twice and each time construct a memory map.
   i) Did the addresses change between runs? Why or why not? Justify your answer.
   j) Provide the valgrind in the PDF.

4. Provided is a C file named cscd340_s18_lab1prob4.c

   a) Within the C file are 9 variable declarations your task is to identify where those variables are located within the process image. The valid location values are: data segment global, data segment local, BSS global, BSS local, heap local, and stack local. Take the code from prob 4 and make your guess.

   b) Using either readelf or objdump verify your answers are correct. If you don't know how to use either execute the command readelf or objdump –help

      If you were incorrect state so and what the correct location is.

   NOTE: It is important that you have a fundamental understanding of the process image, and where items are located, especially the differences between the HEAP and the stack.

5. Provided is a C file named cscd340_s18_lab1prob5.c

    a) I have provided genericArray.h. I have also provided an API specification. To see the methods, you need to write open doc/index.html. In short you need to write:

        1. genericArray.c – Write the methods specified in genericArray.h

        2. myInt.h and myInt.c – Write the structure, the method headers and the method bodies as specified in the API.

        3. myWord.h and myWord.c - Write the structure, the method headers and the method bodies as specified in the API.

    b) I provided a Makefile to compile your code – simply type make

    NOTE: You can't modify the main

TO TURN IN

A zip

- Your PDF file
- All C files
- My Makefile
- A valgrind run for problem 5 - named cscd340_s18_lab1prob5val.txt - showing you are leak free.
- A sample run of problem 5 illustrating everything works – named cscd340_s18_lab1prob5out.txt

Name your zip – your last name first letter of your first name lab1.zip (Example: steinerslab1.zip)

Please keep all the naming lowercase