# Pointers Activity

Pointer Activity I (10 points)

Create a pointer activity source file -  cptr.c - that takes one argument, a number from 1 to 5 - if no arguments are provided then default to 5.

1. Create variables for a character,  an integer, a string and a floating point number.
2. The string variable is a char array of a fixed length.  In other words, the array variable will have to store the string read from console.
3. Define pointers to those variables types without any initialization of those points to the previous variables.
4. Ensure the argument entered is from 1-5 and if no argument is provided default to 5.
5. Ask (i.e. prompt) the user to enter information for each of these variables one at a time. For each variable tell them what they should enter. (For example: “Enter a single character” or “Enter a floating point number” or "Enter an integer" or "Enter a phrase" ) Read in the information and put it in the variables or necessary locations.
6. Print the unassigned pointers. [Note: whatever the pointer is pointing to before initializing]
7. Assign pointers to the corresponding variables.
8. Print out values for the four variables without using the variables themselves. Only use the pointers notation.
9. In addition print out the addresses of the memory locations that each pointer points to
10. Repeat as many times as the argument provided.

Output should look something like:

Character Pointer: XXXXXXXX

Integer Pointer: XXXXXXXX

String Pointer: XXXXXXXX

Floating Point Pointer: XXXXXXXX

Character: h

Integer: 8

String: Carlos is here

Floating Point: 4.5

Character Pointer: YYYYYYY

Integer Pointer: YYYYYYY

String Pointer: YYYYYYY

Floating Point Pointer: YYYYYYY

Test Cases:

.  cptr 2

. cptr 4

. cptr 8

. cptr

.cptr 1 4

Pointer Activity II (10 points)

Create a pointer activity source file -  cptr2.c - that takes two arguments, a number from 1 to 3, and a string sentence(s).

1. Create variables for a character,  an integer, a string pointer.   Based on integer value you will use that number of string pointers.
2. The string variable is a string pointer that has not been allocated.
3. Define pointers to those variables types without any initialization of those points to the previous variables e.g. pChr, pInt, pSentence.
4. Ensure the argument entered is from 1-3 and the sentence are entered.  Errors out if not.
5. Assign: [using argc and argv]
   1. Integer - to number entered from 1-3
   2. Character - to first letter in the sentence.
   3. String pointer(s) - to sentence(s)
6. Assign pointers to the corresponding variables.
7. Call a Function on a separate file -  pointPrint.c -  passing the initialized pointers.   Print out values for the three variables without using the variables themselves as you passed the pointers. Make sure the function also decrements the counter passed by the integer pointer.
8. Repeat as many times as the argument provided.

Output should look something like:

.cptr2   1  help people in need

Character: h

Integer: 1

String: help people in need

 or

.cptr2   2  "what time is it"  "how are you doing"

Character: w

Integer: 2

String: what time is it

Character: h

Integer: 1

String: how are you doing

**Test Cases:**

.cptr2   1  help people in need

.cptr2   2  "what time is it"  "how are you doing"

.cptr2   3 "be kind"  "be good"  "show empathy"

.cptr2   4                                      // Error

.cptr2   2                                      // Error

.cptr2  3 "be kind"                       // Error

Note: make sure to submit your Makefile and source code