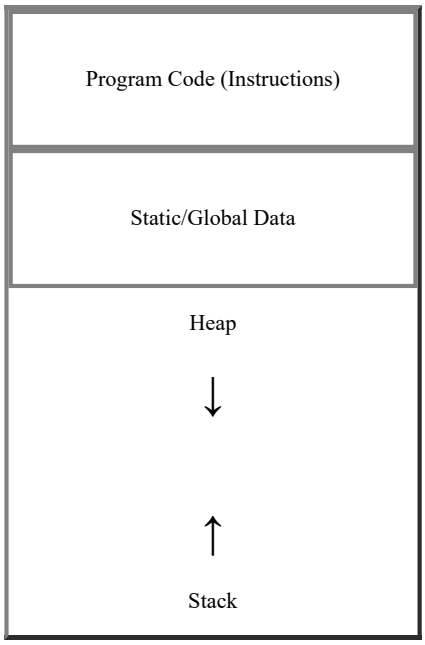
CSC 434

Assignment 5 (C) - Due 4/26/2017 ~~4/25/2017~~

In this assignment, you will write a program in the C language. You will experiment with how data is organized and where it is located within a process' logical memory. A process' logical memory is organized as follows:



You need to write a main program and then several functions. You will also create several types of data and determine the address of your data. To determine the address of a variable, use the ampersand (&) operator. For example, if you declare a variable called x, then you can determine where in logical memory it is located (which will be in hex) by doing:

```
printf ("Address of x = %p\n", &x);
```

However, for dynamically allocated data and literals, you will need to have a variable pointer that points to it. Then to see the address of the data to which the pointer points, ***you won't use the ampersand***. If you use the ampersand, you will get the address of the variable that holds the address, not the address of the data.

Part 1

You need to create the following kinds of data:

1. Parameters
2. Local Variables
3. Static Variables (local not global)
4. Global Variables
5. Dynamically Allocated Data (using malloc or calloc)

For each of these different kinds of variables, you need to declare at least one each of: `int`, `double`, `char`.

In addition to data of the above types, you also should create and discover that address of

1. functions (include the function main())
2. literals

For the dynamically allocated data, you want to print the address of the data as well as the address of the variable that is the pointer to the data. The address of the data to which the pointer points is simply the value stored in the pointer.

You can print the address of a function (the actual code) the same way you print the address of a variable, but without the ampersand or parentheses. In other words, the name of a function is its address. To create and discover the address of a literal, you will need to use a string, because it is difficult to take the address of a literal number. So create a string pointer that points to a string literal in double quotes. Then print the value of that pointer.

In summary, when to use the ampersand and when not to:

Class of Data	Use the ampersand to get the address?

Parameters	Yes
Local Variables	Yes
Static Variables	Yes
Global Variables	Yes
Dynamically Allocated Data (address of the pointer variable)	Yes
Dynamically Allocated Data (address of the data to which the pointer points)	No
Functions	No
String literal	No

Draw a diagram of the process' logical memory (like the picture above) and show approximately where each variable resides. Include the actual addresses (in hex) in your diagram. It does not matter exactly where you place your variables on the picture, as long as you have them in the correct location ***RELATIVE*** to each other. You aren't trying to diagram the entire process' logical memory. You are only drawing that diagram above and labeling some of the data/instructions at approximately where they are.

NOTE: The organization of the segments may be different that the diagram show above, depending on the system you are using. For example, programs generated with Microsoft products seem to have the stack and heap in the middle growing away from each other. Your diagram does not need to be organized exactly as it is in the figure above. However, it ***should*** *have the data in the correct relative position based upon the size of the addresses.*

Part 2

Implement the recursive function factorial(). This needs to be a recursive implementation, not interative.

After you feel that your recursive function is working correctly, the determine the addresses of each instance of the parameter (along with its value) for each recursive call to the function. The initial call to factorial() should be with an argument of 6 (which has 6 recursive calls).

Draw a diagram of the process' stack through the call sequence, including return value, return location, static link, dynamic link, parameters, and local variables. This diagram should contain the addresses (in hex) of the parameter and its value. Remember that these fields on the stack are not single bytes; and some of them may not be used. Using just the addresses of the parameters, how big is the activation record for this function? Remember that the addresses refer to *words,* not *bytes* or *bits.*

Part 3

Create a 3-dimensional array of doubles with sizes [4][2][3]. Using 3 nested loops, iterate through every instance of the array and determine the address of each location along with the array subscripts.

Create a diagram showing the addresses of all 24 elements of the array. Determine the formula, based upon the formulas we discussed in Chapter 6 (row-major and column-major), that will map the values of the array subscripts to the address of each instance. This formula should be very specific, including values specific to the above array (as in, it should use the values in the declaration: 4, 2, and 3 as well as i, j and k). Does C use row-major or column-major ordering?

Documentation:

It is expected that you will follow standard practices of documentation of your program. That means that classes and methods should have header information include: Author, date written, list and description of parameters or data members (where appropriate), type and description of return values (where appropriate), a general description of the purpose of the class or method. You should also put in comments for code for which its meaning is not obvious. That does not mean to put in comments that a loop will loop through some values, but rather put in comments for things that it would take study or searching documentation to understand. You should also put in comments for constructs that are not common in every language (e.g. use of regular expressions). And finally, any part of the program that asks the user for input should be proceeded with an appropriate prompt. An appropriate prompt is one that tells the user what to enter and in what form. For example, "Enter two floating-point numbers separated by whitespace on one line."

Deliverables:

- The source files for the program
- Explain the features you used in the program. In particular, explain what the language constructs, libraries, and functions do.
- *The diagrams you drew plus the answers to any questions above.*
- Submit the files to the assignment drop box on Blackboard. You can get there by logging into UNCW SeaPort, then clicking on the link for the course.

This page was last updated: **April 18, 2017**

Email: cferner@uncw.edu