Lab 2

Omar Raza

Due: 6/28/2016

Problem 1:

a) When dealing with variables and variable decelerations, it is always the matter of two things, if you are using the variables content, or the variables address in memory. Therefore, when using the **stdio.h** function, "scanf", programmers have to be equally cautious because of the easiness to make a mistake. We can look at two pieces of code, and describe how one is correct, and one is wrong. The first function,

scanf("%d",s);

is using the stdio.h function, "scanf", to input a value and place it as the **address** of variable s. After doing that, scanf will then try to place the value, and will fail causing in a segmentation fault, which I will discuss in 1.b. The next and proper line of code is:

scanf("%d",&s);

It is using the stdio.h function, "scanf", to input a value and point it to the spot in memory where "s" is stored, and store it.

b) A segmentation fault is thrown by the gcc when a program tries to access a restricted or non-existent place in memory.

Problem 2:

a) All I did in main.c of v1 was change all the occurrences of s to the int pointer *u, that we were told to create. This works because of pointers, where instead of using the variable itself, I pointed to the location of it in memory.

Problem 3:

Available in the v2 directory of the lab2 folder

Problem 4:

a) If v8 is left alone as is and compiled, then it will result in a segmentation fault because the place in memory 'h' is accessible, but the places in memory 'h+1' and 'h+2' either do not exist or are illegal to access.

b) Debugged in v6 of lab2

c) The fundamental difference between module v7 and v8 is the declaration and input of the values, v7 relies on array inputing

h[1] = 4;

While v8 uses pointers to place values in the array

*(h+1) = 200;

d) The way to fix v8 is to declare an array of length 3 so that it is able to accommodate all the declaration:

int h[3];

Problem 5:

a) A silent run-time error is an error that doesn't occur in syntax or usage, something that would be caught during compile time, but the logic of the program itself, that is manifested only during run-time. The silent run-time error has to deal with the for loop statements. If you change the 6 to a 7 in the loop parameters, it throws a very small number because of the undefined nature of the 7th part of the array.

b)One way to not have any errors of decrease the risk of errors is to make compound for-loop statements that wont need to be redeclared every time one wants to do something else with the array.

Problem 6:

Available in the v10 directory of the lab2 folder

Bonus Problem:

a) The program will crash.

b) Because when inputing a string longer than the character array, it tries to access spaces in memory that aren't available. So the program crashes and aborts.