**Think-Pair-Share 2**

1. **How do you compile your** punishment.c **so that you can debug it using GDB?(make the executable call** punish**)**

**gcc -g -o punish punishment.c**

1. **Once punishment.c is compiled, how do you load it in GDB? Try it with your program.**

**gdb ./punish**

1. **Once punish is loaded, how do you run it in GDB? Try to run your punish.**

You type **run** or **r**

1. **What are breakpoints? How do you set a breakpoint at a certain line of your program? Try to set a breakpoint in punishment.c where the for loop begins.**

Breakpoints are[...]. To set a breakpoint you type **b<line>** and the line number where it will break. My for loop begins at line 17 so it will be **b 17.**

1. **Now run the program again. It should stop at the breakpoint you set in Q4. From here, how do you run the program line by line? Try to run the next 3 lines with your program.**

You type in **s** to run line by line.

1. **While you are still running punish line by line, how can you see the value of a variable? Pick 3 variables in your program and display them in the terminal one by one.**

**p <nameOfVariable>** or **print <...>.**  You have to be in the specific line of the variable.

1. **Now you are tired of running line by line. How do you let the program finish its run? Try to finish running your punish.**

You type in **c** or **continue**.

1. **How do you exit from GDB?**

You type in **q** or **quit** exit.

**Think-Pair-Share 3**

**1. How many variables were declared in line 1? How many of them are . . .pointers(and what are they)?**

4 variables were declared in line 1. 2 pointers called **\*px** and **\*py**.

**2. What will be the values of x, y, and arr[0] if you run the program? . .Validate your answer by running the program. Why do you think it . . happens that way?\You will need to insert print statements to . . . . display those values.**

The values I received from printing were:

**printf("%d\n", x);**  = -1276885139

**printf("%d\n", y);** = 21998

**printf("%d\n", arr);** = -621590112

This happens because these variables are initialized to garbage.

**3. How do you prevent x, y, and the content of arr from having . . . . . .unexpected values? Try to fix them in the program.**

Just initialize them to whatever value you need value.

**4.The moment you have declared a variable, the program will allocate a . memory location for it. Each memory location has an address. Now . . . insert print statements to display the addresses of x, y.**

**printf("%p\n", &x);** = **0x7ffce475e8c8**

**printf("%p\n", &y);** = **0x7ffce475e8cc**

**5. Now insert code so that px points to x and py points to y. Print . . . .out the values and addresses of those pointers using only the . . . . .pointer variables (yes, pointers have addresses too!) You should . . .see that value of px equals to address of x, and the same is true . . .with py and y.**

**x = 10; // I set y and x to 10 so that I could tell the . . . . .y = 10 // difference between the print statements**

**px = &x;**

**py = &y;**

**printf("%p\n", px); //address = 0x7ffd544d2168**

**printf("%u\n", \*px); //value = 10**

**printf("%p\n", py); //address = 0x7ffd544d216c**

**printf("%u\n\n", \*py); //value = 10**

**6. As we’ve learned in lectures, an array name can be used as a . . . . .pointer to access the content of the array. Write a loop to print . . .out the content of arr by using arr as a pointer (do not use []).**

**for(int i = 0 ; i <= 9 ; i++){**

**printf("%d\n", \*arr);**//prints out the value of arr

**\*arr += 4;**//adds four bytes for integers

**}**

**7. Are array names really the same as pointers? Let’s find out! An . . .. array name points to the first element of an array, so arr should . . .point to the address of arr[0]. Insert code to verify this.**

**printf("%p\n", &arr);** = 0x7fff0d93a740 //address of arr

**printf("%p\n", &arr[0]);** = 0x7fff0d93a740 //address of arr[0]

**8. Now print out the address of arr. Does the result make sense? Why?**

**printf(“%d”, \*arr); =** 0x7fff0d93a740

/\*The result makes sense since it is pointing at the specific memory location and it hasn’t changed\*/

**Segmentation fault**

1. **What line caused the segmentation fault?**

The segmentation fault was caused by multiple lines:

**(gdb) bt**

**#0 0x00007ffff7a538c2 in \_IO\_vfscanf\_internal (s=<optimized out>,**

**format=<optimized out>, argptr=argptr@entry=0x7fffffffddc0,**

**errp=errp@entry=0x0) at vfscanf.c:1898**

**#1 0x00007ffff7a5ffd8 in \_\_isoc99\_scanf (format=<optimized out>)**

**at isoc99\_scanf.c:37**

**#2 0x00005555555547f2 in read\_values (sum=1) at average.c:15**

**#3 0x0000555555554835 in main () at average.c:23**

The lines are **23** in main, where the **int read\_values(){..}** function is called, and **15** in the while-loop where we print **scanf(“%d”, input);** this is expecting an argument of **int \*** but it’s taking an **int**.

1. **How do you fix the line so that it works properly?**

In order for it to work properly,we have to change it to **scanf(“%d”, &input);**

1. **What is the bug here?**

This is the situation, when we call the function

**values = read\_values(sum)** we are sending a clone of **sum** to our **read\_values(){..}** function, which disappears after the function ends.

1. **How do you fix it?**

To fix it we have to dereference the address of **sum** so that when we call **values = read\_values(&sum)** we can send the memory address to the function so that when the value of that pointer **\*sum** is changed in the function, **sum** in **int main(){..}**  will change as well.

**Fix appendTest.c**

1. **Run the program with the following input: *“HELLO!”* for** str1 **and *“hello!”* for** str2**. Is the output expected?**

The output is not expected, it prints out *“****HELLO!hello!llo!”***

1. **Do not stop the program, enter *“HI!”*  for** str1**, and *“hi!”* for** str2**. Is the output expected? What is the bug here? Try to fix the program so that it will print correctly**

The output is not expected again, this time around it prints ***“HI!hi!hellhi!”*** which includes part of the string that was stored last time it asked for an input.

1. **Do not stop the program, enter *“Hello!, how are you?”* for** str1 **and *”I am fine, thank you!”* for** str2**. Is the output expected? Why do you think this happens?(You don’t need to fix this)**

*The output is* ***“Hello! HowI am fine, thank you!I am fine, thank you!.”*** and this is not what is expected. I think it is an issue with the length of **s1len** and **s2len** there is something messed up when you call these into the for loop.