I'm getting seg faults in all of my Comparator functions and I can't put my finger on why. Of course I'm using the typedef in the spec, so all my comparator function definitions follow it. I am using the comparator found in integer.c (provided by Dr. Lusth) as a basis for my comparator functions for int and real (instead of using integer *, I use real * for real). Below is the comparator function found in integer.c:

```c
int compareInteger(void *v, void *w)
{
    return ((integer *) v)->value - ((integer *) w)->value;
}
```

I don't see any problem with how that works. In my sort function, I'm simply using something like:

```c
if(comp(peekQueue(input), peekQueue(output)) <= 0)
    // do blah blah blah
```

It reaches the comparator function, and then seg faults. I am grateful for any suggestions.

---

Subject: Re: Seg Faulting in my Comparator functions
Posted by davidmccoy on Wed, 08 Feb 2017 18:29:26 GMT

Guessing that maybe your queue doesn't have anything left in it, or output doesn't yet have anything in it, comp tries to peek, and then it seg faults.

---

Subject: Re: Seg Faulting in my Comparator functions
Posted by cewrobel on Wed, 08 Feb 2017 18:33:11 GMT

davidmccoy wrote on Wed, 08 February 2017 12:29: Guessing that maybe your queue doesn't have anything left in it, or output doesn't yet have anything in it, comp tries to peek, and then it seg faults.

That wasn't what I was literally doing, just an example. Didn't want to get in trouble for posting actual code haha. It segfaults on my comparison between the dequeued value and the next value to be dequeued in input.
Unless there's a check to not let it peek on empty input (which happens right after the last value is dequeued) that could be the problem.

I personally can't think of another reason why it's messing up without knowing if you typed the compareIntegers function correctly in comparator.c, but even if you did have an error there, that probably wouldn't result in a seg fault being the error.

I had this issue... When I was on the last element of the input queue, it would try to grab peekInput after I dequeued and nothing would be there. Resulting in a seg fault because it's comparing the dequeued value and null.

You want to check the size prior to trying to peek/dequeue. It's seg faulting when you try to peek/dequeue from an empty queue more than likely. Something like this would make sure you don't

if ((sizeQueue(input) != 0) && (sizeQueue(output) != 0) && (comp(peekQueue(input), peekQueue(output)) <= 0))
   //do blah blah blah

I check to see if the size of input is 0 or 1 at the top of my sorting function. Also, it's segfaulting on the first dequeued item, so I know the queue has a good couple more items in the list.

Also, the example of how I'm using it in the original post is not a real if-statement, it was just an example of how I'm calling comp.
The only reason your comparator would segfault is if it is passed a null pointer. Check your queue and sll code.