Interactive Debugger

This Python program contains a function to compute the largest number in a list of numbers, as well as a number of test cases, several of which fail.

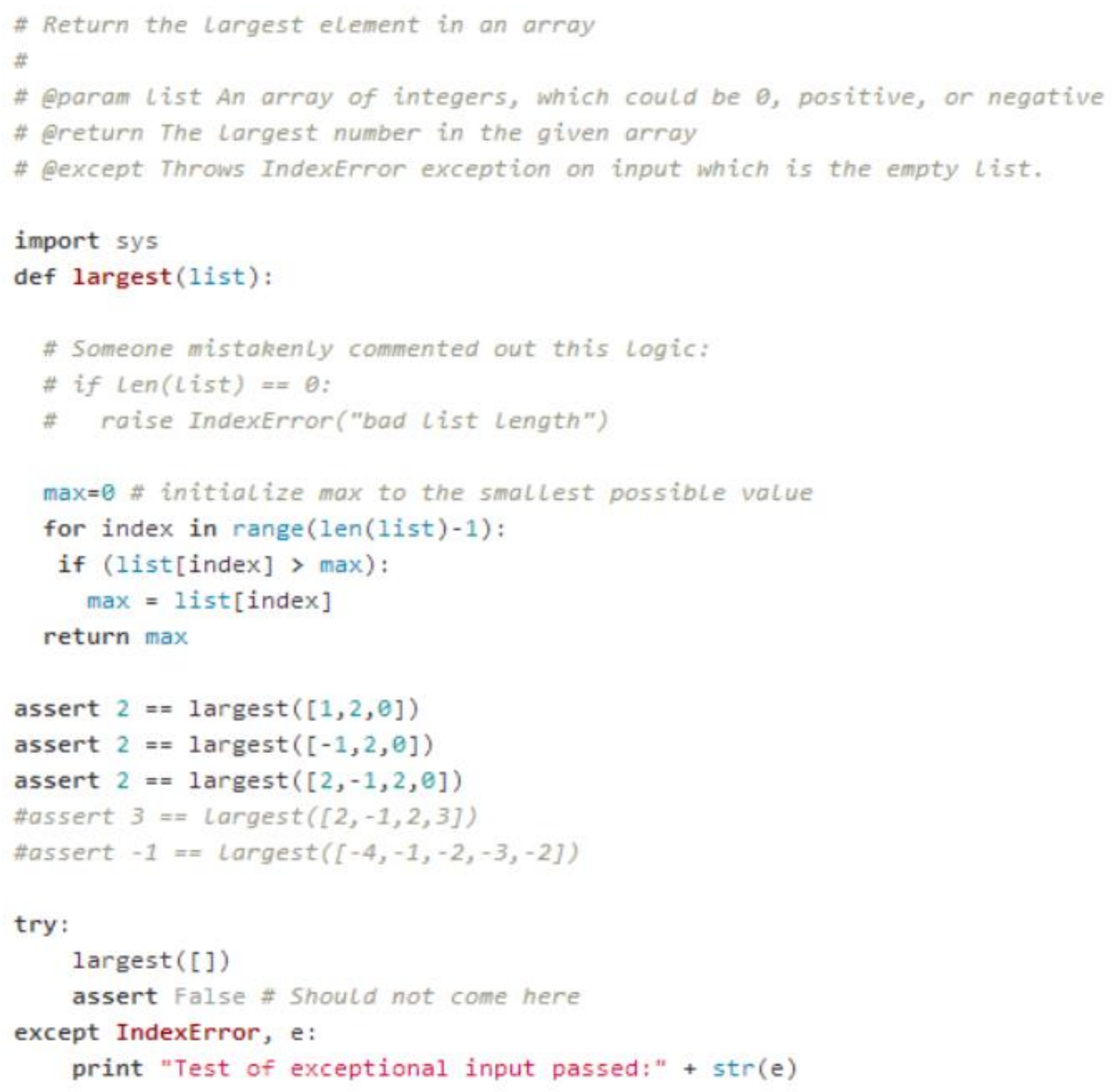
Use the debugger to debug these test cases. Use the break (both at a function and at a specific line), print, step, continue, and finish commands to debug.

Explain how you proceeded.

Clearly, you can eyeball this example, but you are required to use the debugger.

Solution:

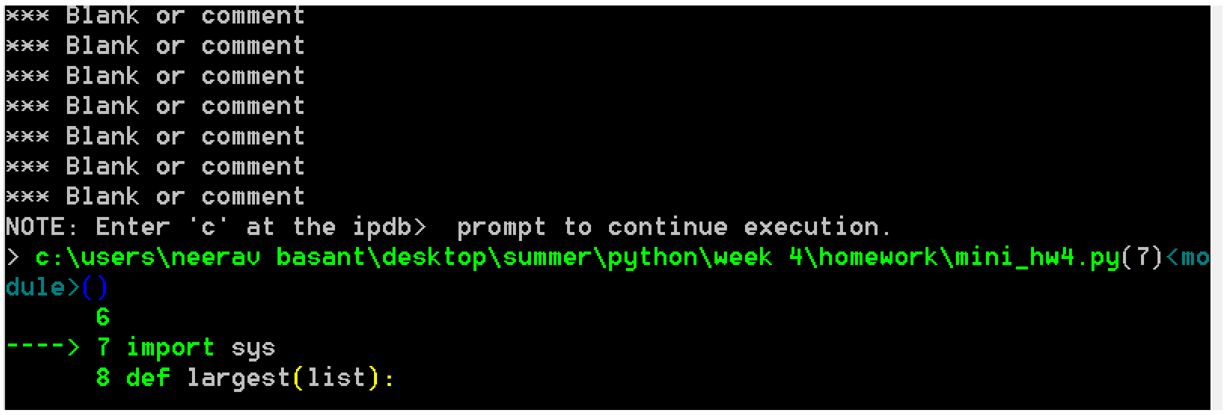
Python Program –



```
# Return the largest element in an array
#
# @param list An array of integers, which could be 0, positive, or negative
# @return The largest number in the given array
# @except Throws IndexError exception on input which is the empty list.

import sys
def largest(list):

   # Someone mistakenly commented out this logic:
   # if len(list) == 0:
   #    raise IndexError("bad list length")

   max=0 # initialize max to the smallest possible value
   for index in range(len(list)-1):
    if (list[index] > max):
      max = list[index]
   return max

assert 2 == largest([1,2,0])
assert 2 == largest([-1,2,0])
assert 2 == largest([2,-1,2,0])
#assert 3 == largest([2,-1,2,3])
#assert -1 == largest([-4,-1,-2,-3,-2])

try:
    largest([])
    assert False # Should not come here
except IndexError, e:
    print "Test of exceptional input passed:" + str(e)
```

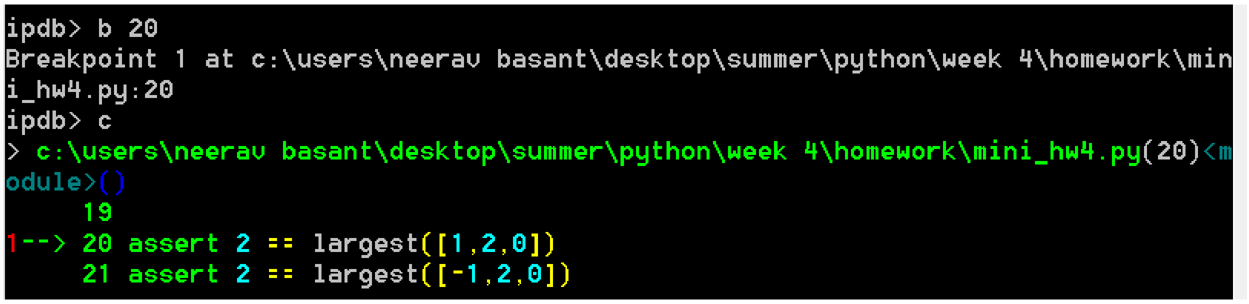
Step 1: Run the code using IPython debugger

%run –d Mini_HW4.py

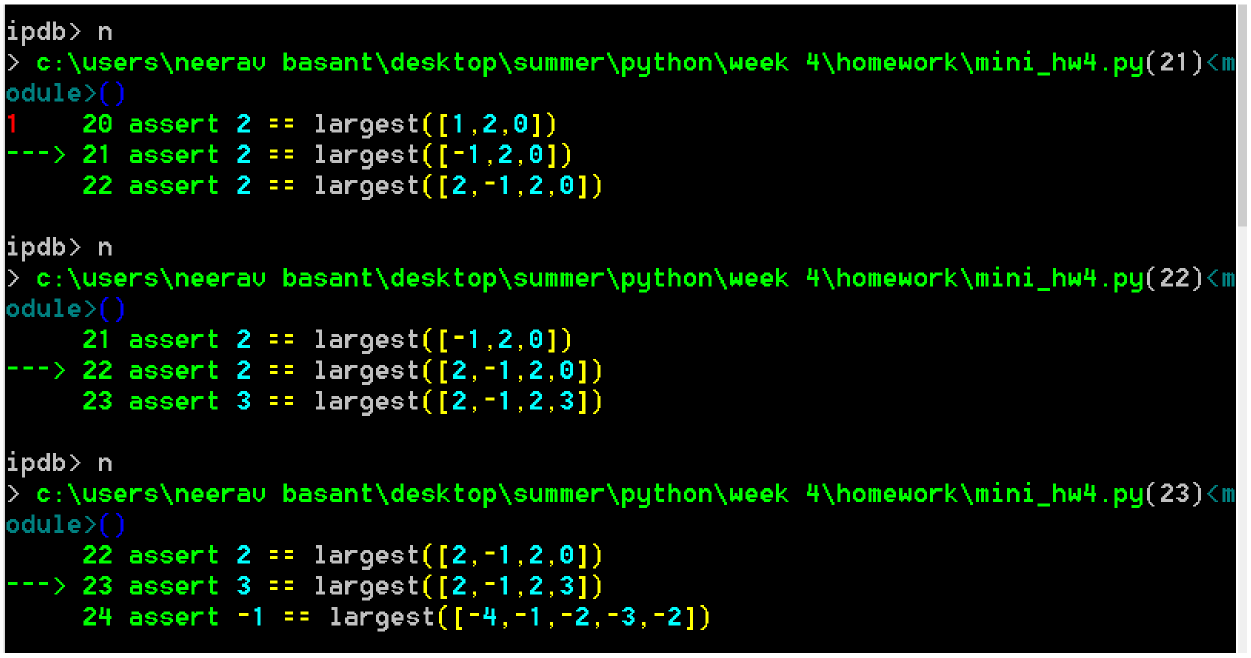


```
*** Blank or comment
*** Blank or comment
*** Blank or comment
*** Blank or comment
*** Blank or comment
*** Blank or comment
*** Blank or comment
NOTE: Enter 'c' at the ipdb>  prompt to continue execution.
> c:\users\neerav basant\desktop\summer\python\week 4\homework\mini_hw4.py(7)<mo
dule>()
      6
----> 7 import sys
      8 def largest(list):
```

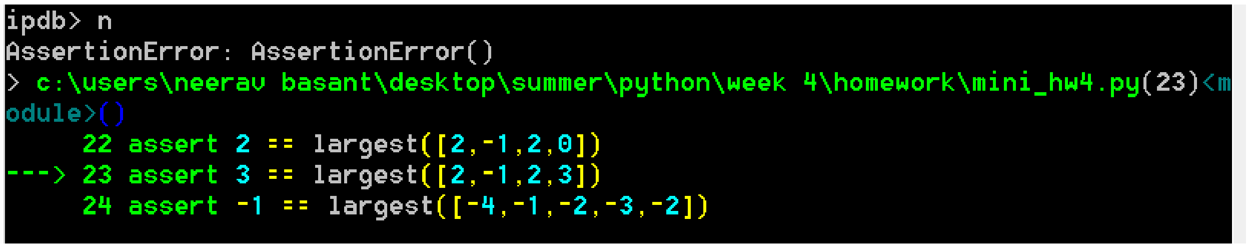
Step 2: We will set the break points at all the assert statements and make sure we pass all the test cases by correcting the code. We will also uncomment the two test cases (assert statements) that are currently commented.



```
ipdb> b 20
Breakpoint 1 at c:\users\neerav basant\desktop\summer\python\week 4\homework\min
i_hw4.py:20
ipdb> c
> c:\users\neerav basant\desktop\summer\python\week 4\homework\mini_hw4.py(20)<m
odule>()
     19
1--> 20 assert 2 == largest([1,2,0])
     21 assert 2 == largest([-1,2,0])
```

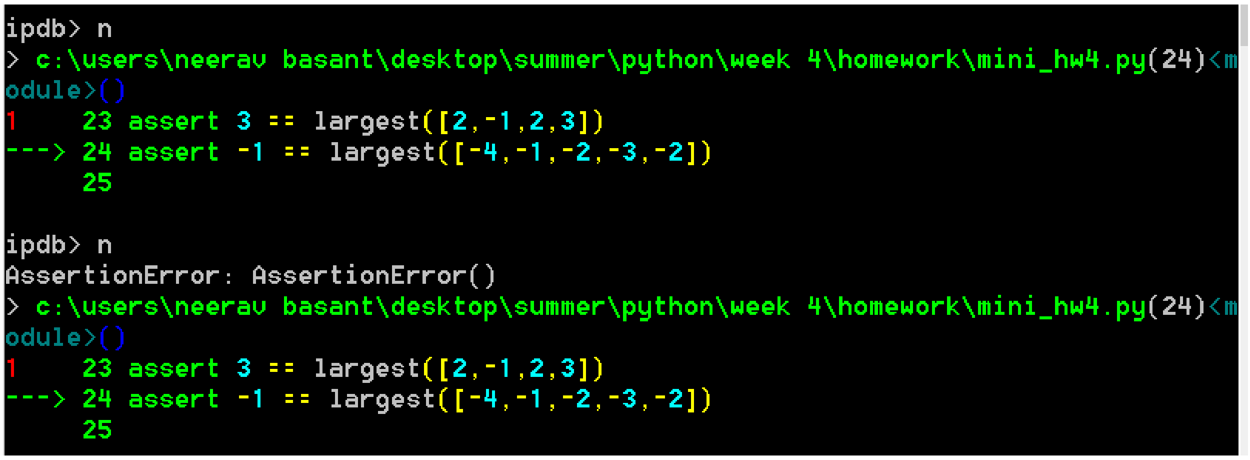


```
ipdb> n
> c:\users\neerav basant\desktop\summer\python\week 4\homework\mini_hw4.py(21)<m
odule>()
1    20 assert 2 == largest([1,2,0])
---> 21 assert 2 == largest([-1,2,0])
     22 assert 2 == largest([2,-1,2,0])

ipdb> n
> c:\users\neerav basant\desktop\summer\python\week 4\homework\mini_hw4.py(22)<m
odule>()
     21 assert 2 == largest([-1,2,0])
---> 22 assert 2 == largest([2,-1,2,0])
     23 assert 3 == largest([2,-1,2,3])

ipdb> n
> c:\users\neerav basant\desktop\summer\python\week 4\homework\mini_hw4.py(23)<m
odule>()
     22 assert 2 == largest([2,-1,2,0])
---> 23 assert 3 == largest([2,-1,2,3])
     24 assert -1 == largest([-4,-1,-2,-3,-2])
```



```
ipdb> n
AssertionError: AssertionError()
> c:\users\neerav basant\desktop\summer\python\week 4\homework\mini_hw4.py(23)<m
odule>()
     22 assert 2 == largest([2,-1,2,0])
---> 23 assert 3 == largest([2,-1,2,3])
     24 assert -1 == largest([-4,-1,-2,-3,-2])
```

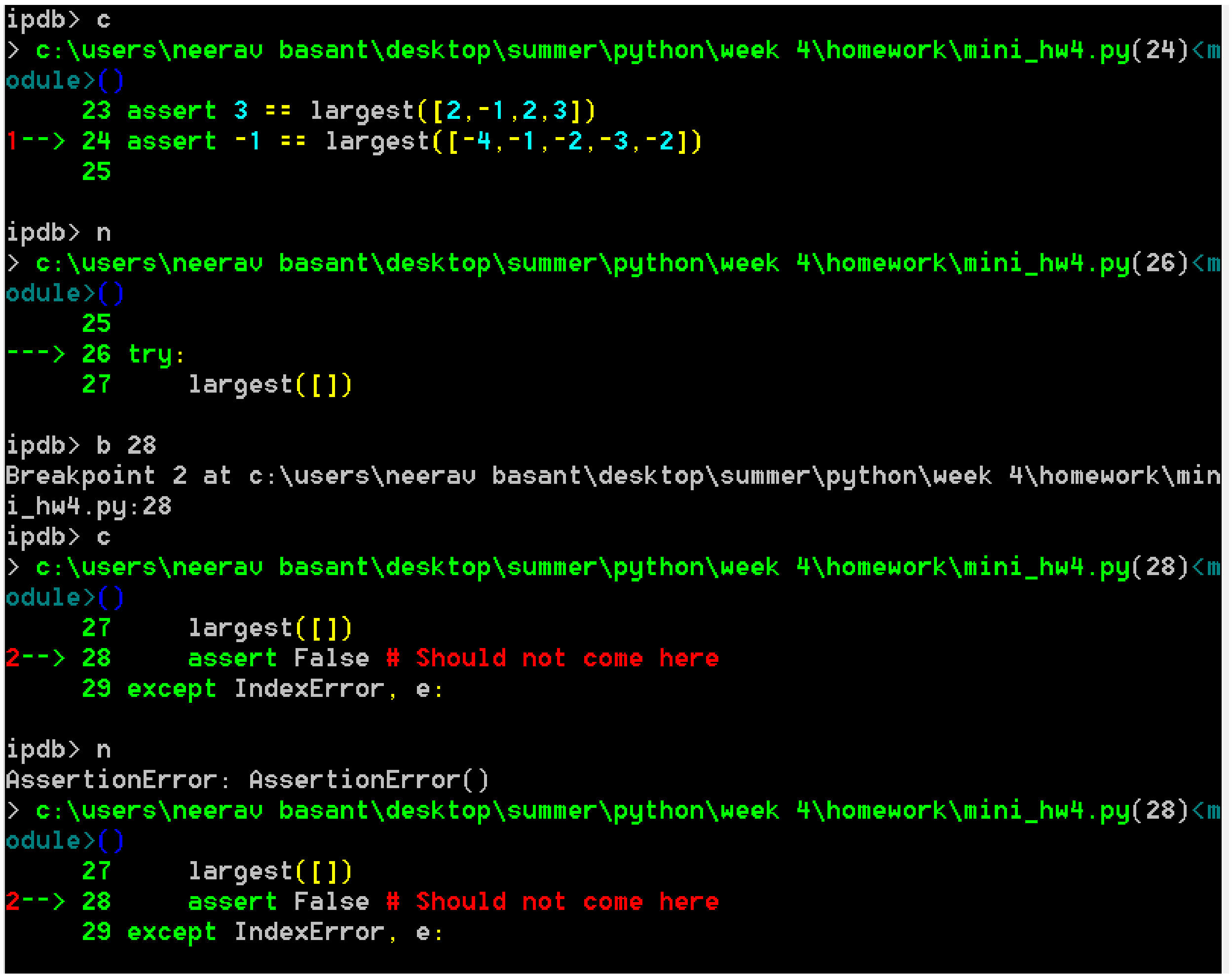
Step 3: At line 23 we got an assertion error. This error was due to the fact that ‘for’ loop is not including the last element of the list. We can modify the range function and rerun the code.



```
ipdb> n
> c:\users\neerav basant\desktop\summer\python\week 4\homework\mini_hw4.py(24)<m
odule>()
1    23 assert 3 == largest([2,-1,2,3])
---> 24 assert -1 == largest([-4,-1,-2,-3,-2])
     25

ipdb> n
AssertionError: AssertionError()
> c:\users\neerav basant\desktop\summer\python\week 4\homework\mini_hw4.py(24)<m
odule>()
1    23 assert 3 == largest([2,-1,2,3])
---> 24 assert -1 == largest([-4,-1,-2,-3,-2])
     25
```

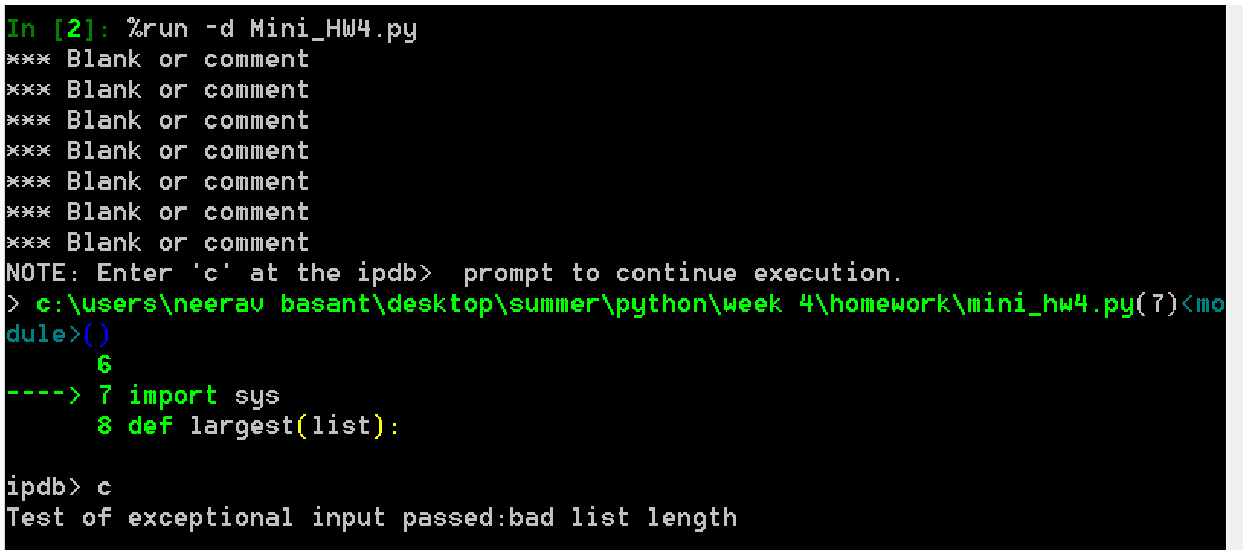
Step 4: At line 24 we again got an assertion error. This error was due to the fact that all the elements in the list were negative and we had set the max = 0.

We can modify the code by setting the value of max to the minimum possible integer value i.e. **max = - sys.maxint -1**



```
ipdb> c
> c:\users\neerav basant\desktop\summer\python\week 4\homework\mini_hw4.py(24)<m
odule>()
     23 assert 3 == largest([2,-1,2,3])
1--> 24 assert -1 == largest([-4,-1,-2,-3,-2])
     25

ipdb> n
> c:\users\neerav basant\desktop\summer\python\week 4\homework\mini_hw4.py(26)<m
odule>()
     25
---> 26 try:
     27     largest([])

ipdb> b 28
Breakpoint 2 at c:\users\neerav basant\desktop\summer\python\week 4\homework\min
i_hw4.py:28
ipdb> c
> c:\users\neerav basant\desktop\summer\python\week 4\homework\mini_hw4.py(28)<m
odule>()
     27     largest([])
2--> 28     assert False # Should not come here
     29 except IndexError, e:

ipdb> n
AssertionError: AssertionError()
> c:\users\neerav basant\desktop\summer\python\week 4\homework\mini_hw4.py(28)<m
odule>()
     27     largest([])
2--> 28     assert False # Should not come here
     29 except IndexError, e:
```

Step 5: At line 28 we got an error. This was due to the fact that we are passing an empty list and the asserting false. If we don’t want to run the start statement, we need to catch this exception before that statement runs. But unfortunately, the code that was supposed to perform this function (line 11 -12) is commented. We will uncomment that part of the code and run the whole code again.



```
In [2]: %run -d Mini_HW4.py
*** Blank or comment
*** Blank or comment
*** Blank or comment
*** Blank or comment
*** Blank or comment
*** Blank or comment
*** Blank or comment
NOTE: Enter 'c' at the ipdb>  prompt to continue execution.
> c:\users\neerav basant\desktop\summer\python\week 4\homework\mini_hw4.py(7)<mo
dule>()
      6
----> 7 import sys
      8 def largest(list):

ipdb> c
Test of exceptional input passed:bad list length
```

Step 6: Since the code ran without error and exited the debugger, we have debugged the code completely. Please find below the updated code:

```python
# Return the largest element in an array
#
# @param list An array of integers, which could be 0, positive, or negative
# @return The largest number in the given array
# @except Throws IndexError exception on input which is the empty list.

import sys
def largest(list):

  # Someone mistakenly commented out this logic:
  if len(list) == 0:
    raise IndexError("bad list length")

  max= -sys.maxint - 1 # initialize max to the smallest possible value
  for index in range(len(list)):
   if (list[index] > max):
     max = list[index]
  return max

assert 2 == largest([1,2,0])
assert 2 == largest([-1,2,0])
assert 2 == largest([2,-1,2,0])
assert 3 == largest([2,-1,2,3])
assert -1 == largest([-4,-1,-2,-3,-2])

try:
    largest([])
    assert False # Should not come here
except IndexError, e:
    print "Test of exceptional input passed:" + str(e)
```