**Exercise on Code Tracing Examples**

Let’s use this nested loop as an example of how to do a code trace. We will be using these often when we get to searches and sorts.

**for(int x=1; x<7; x++ )** // TraceCodeExample.java

{

**for(int y=0; y<x; y++ )**

{

System.out.print(" a b c ");

}

System.out.println(" dog ");

}

OUTPUT:

a b c dog

a b c a b c dog

a b c a b c a b c dog

a b c a b c a b c a b c dog

a b c a b c a b c a b c a b c dog

a b c a b c a b c a b c a b c a b c dog

With a code trace or walkthrough, the person doing the trace “pretends” to be a computer and mimics the computer’s steps. The hardest part about doing a trace is being “as dumb as a computer”. As people we are very smart and can see the big picture. Computers can only recognize whether one circuit at a time is on or off – I guess you could say they have tunnel vision. So when you do a trace, don’t look ahead or you can really get lost once we get to some of the complicated sorts.

First Step in a trace – on a chart make a “memory area” for each variable and for each comparison.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **x** | **x<7** | **y** | **y<x** | **output** |
| 1 | 1<7 true | 0 | 0<1 true | a b c |
|  |  | 1 | 1<1 false | dog then newline |
| 2 | 2<7 true | 0 | 0<2 true | abc |
|  |  | 1 | 1<2 true | abc |
|  |  | 2 | 2<2 false | dog then newline |
| 3 | 3<7 true | 0 | 0<3 true | abc |
|  |  | 1 | 1<3 true | abc |
|  |  | 2 | 2<3 true | abc |
|  |  | 3 | 3<3 false | dog then newline |

**Note that we did not put a number in x unless the computer did. That is why the shaded squares don’t have any thing in them.** The computer only puts something there if there is an assignment statement with x as an lvalue. When we do a trace we don’t just do it the way we want, we only do what the computer does. We must have tunnel vision. That becomes very important on some sorts such as the shell sort. Also, **notice that on traces you do for me you can’t just put true or false, you have to put what is true or false – ex. 2<3 true**

You should be able to pick up some insight about how nested loops work by doing the sort. Notice that the inner loop “starts over” every time we test the outer loop and it is *true*. In other words if **x<7** is *true*, then **y** is set back to *0*. Why is string “**a b c”** printed once more with each line? How many times will *x* be initialized? How many times will *y* be initialized? Notice that the output dog statement is not part of the inner loop, and it is actually part of the outer loop.

On the table below, finish working through the code until the outer loop terminates.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **x** | **x<7** | **y** | **y<x** | **output** |
| 1 | 1<7 true | 0 | 0<1 true | a b c |
|  |  | 1 | 1<1 false | dog then newline |
| 2 | 2<7 true | 0 | 0<2 true | abc |
|  |  | 1 | 1<2 true | abc |
|  |  | 2 | 2<2 false | dog then newline |
| 3 | 3<7 true | 0 | 0<3 true | abc |
|  |  | 1 | 1<3 true | abc |
|  |  | 2 | 2<3 true | abc |
|  |  | 3 | 3<3 false | dog then newline |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

You can try running program ***TraceCodeExample*** and see the results.

Try modifying *TraceCodeExample.java* and make it printout it’s own “trace” - Can you modify the code to get this output.

1. 0

1

1. 0

1

2

1. 0

1

2

3

….

so on and so forth.