1. Here are the stack data contents.

   (i)  \( a = 2, \ b = 20 \)
   (ii) \( a = 2, \ b = 20, \ r = 3, \ q = 3 \)
   (iii) \( a = 2, \ b = 20, \ r = 3, \ q = 3, \ s = 3, \ t = 10, \ p = 6 \)
   (iv) \( a = 2, \ b = 20, \ r = 3, \ q = 7 \)
   (v)  \( a = 2, \ b = 8 \)

2. (a) Under call-by-value, `twiddle` has no effect on \( p_0 \) or \( p_1 \), so the first two outputs are “0 1”; `swizzle` actually exchanges the contents of the \( a \) fields, so the second two outputs are “1 0”.

   (b) Under call-by-reference, `twiddle` actually exchanges the values of \( p_0 \) and \( p_1 \), so the first two outputs are “1 0”; `swizzle` exchanges things as before, so the second two outputs are “0 1”.

   (c) Assuming un-boxed semantics, the parameters to `twiddle` and `swizzle` are copied when they are passed, creating new objects, and similarly for the assignment to \( z \) in `twiddle`. This has no effect on the visible behavior of `twiddle`, so the first two outputs are again “0 1”. But `swizzle` now operates on local copies of its arguments, so it has no effect on the variables in `main`, and the second two outputs are also “0 1”.

CS558 Programming Languages – Winter 2022 – Suggested Study Question Solutions for Lecture 4b