Wes Rupert – 3/25/13
EECS 345 – Written Assignment 02

1)
- a) {1, 2, 3, 4, 5}
- b) {1, 2, 3, 4, 5}
- c) {1, 2, 2, 4, 5}
- d) {1, 2, 2, 4, 5}

2)
- a) {1, 2, 3, 4, 5}
- b) {1, 2, 6, 4, 5}
- c) {1, 2, 6, 4, 5}
- d) error (save=5 (2 + list[2]), list[5] is out of bounds)

3)
- a) A, B, C, D
- b) A, B
- c) A, B, C

4) I believe it would be equally good or bad to implement either method. It would be good to use reference counters, since the memory is otherwise being managed by the programmer, and not the program. Since the programmer is using pointers to create and destroy memory at will, it would be better if the tombstones were removed as soon as there were no longer any references to them. However, this would incur even more additional memory used per block allocated, as there's already a memory penalty for using tombstones in the first place. On the other hand, mark and sweep would not incur this additional memory overhead, but it would require halting the program to clean up the tombstones, and it is near impossible to tell whether there is still a pointer to the tombstone, especially if the programmer is actively trying to circumvent the tombstones. Overall, if there's gong to be a garbage collector for the tombstones, there might as well just be a garbage collector for the memory, and abandon pointers altogether.

5) If a language uses pass-by-reference, any formal parameters that are used can be modified to create functional side effects, as the function is modifying the outside environment.