Homework 03 – Allie Duncan and Kyle McConnaughay

1. ✓
2. ✓
3. See Appendix A
4. The results from the run are shown below:
   1st run:
   Address of local is 140736555251916
   Address of global is 6295608
   Address of main is 4195763
   Address of r1 is 19337232
   Address of a is 140736555235872
   Address of rv is 19337264
   Address of a is 140736555219824
   Address of rv is 19345280

   2nd run:
   Address of local is 140735124446572
   Address of global is 6295608
   Address of main is 4195763
   Address of r1 is 36118544
   Address of a is 140735124430528
   Address of rv is 36118576
   Address of a is 140735124414480
   Address of rv is 36126592

   3rd run:
   Address of local is 140733643680684
   Address of global is 6295608
   Address of main is 4195763
   Address of r1 is 17149968
   Address of a is 140733643664640
   Address of rv is 17150000
   Address of a is 140733643648592
   Address of rv is 17158016

   As you can see, not everything gets allocated in the same place each time. Variables that are constant or global (like main and global) are allocated the same. But variables stored in the stack and heap are re-assigned to different places each time.
5. The stack grows down, as can be seen from the fact that local address goes from 140736555251916 to 140733643680684. Between each run of

recurse, 16048 bytes are allocated for each instance (gotten from subtracting sequential runs of rv and/or a).

6. The heap goes up and down (as can be by examining r1's position between runs. Between the 1st run and the 2nd, there are -16781312 bytes. However, between the 2nd run the 3rd, there are 18968576 bytes.
7. Two concurrent run of recurse (using a child and parent thread) gives us:

    Parent execution:
    Address of a is 140736798094880
    Address of rv is 24269136

    Child execution:
    Address of a is 139637042659376
    Address of rv is 139636902987968

    Parent execution:
    Address of a is 140736798078832
    Address of rv is 24277152

    Child execution:
    Address of a is 139637042643328
    Address of rv is 139636902995984

    We know that a is in the stack and that rv is in the heap. Therefore, for the parent thread, the stack is around 140736798078832. For the child thread, the stack is around 139637042643328. The fact that concurrent runs of child and parents have such different locations in the stack means that each thread has its own stack. The threads do not share a stack. On the other hand, however the heap does not have any order in removal and assignment. So asking "where" we are in the heap – and printing the address of rv (which is in the heap) doesn't tell us much. It is still undetermined whether the child and the parent share a heap – although we are expecting them to share one.

Appendix A

Kyle McConnaughay
Allie Duncan

HW03 → Question 3

Stack
local: 140,736,555,251,916 → changes between runs
a: 14,073,655,525,872 → changes between runs

Heap
r1: 19,337,232 → changes between runs
rv: 19,337,264 → changes between runs

Globals
global: 6,295,608 → stays the same between runs

Constants

Code
main: 4,195,763 → stays the same between runs