COMP 141: Basic Semantics — Part 1

*Instructions:* In this exercise, we are going to static and dynamic scoping in C.

1. Consider the following program.

#include <stdio.h>

|  |  |
| --- | --- |
| int a = 3; | //p1 |
| int b = 5; | //p2 |
| int f (int b){ return a + b; | //p3 |
| } | //p4 |
| int main () { printf("1.%d\n", b); | //p5 |
| int b = a; | //p6 |
| int a = 2 \* b;  { printf("2.%d\n", b); | //p7 |
| int b = 1;  printf("3.%d\n", b); | //p8 |
| } printf("4.%d\n", b);  b = f (b);  printf("5.%d\n", b); return 0;  } | //p9 |

* 1. For each line marked with //p#, define the symbol table if static scoping is used[[1]](#footnote-1).

For example, in p1, p2 and p3 the tables are as follows[[2]](#footnote-2).

p1.

* + 1. : int global

p2.

a : int global b : int global

p3:

a : int global b : int local to f :: int global f : int -> int function

* 1. According to the static scoping what would be the output of the program (i.e., the prints in standardoutput)?

1. Consider the same program with different line marks. The line marks follow the order in which program is executed.

#include <stdio.h>

int a = 3; int b = 5;

|  |  |
| --- | --- |
| int f (int b){ | //p6 |
| int c = a + b;  return c; | //p7 |
| } | //p8 |
| int main () { printf("1.%d\n", b); | //p1 |
| int b = a; | //p2 |
| int a = 2 \* b;  { printf("2.%d\n", b); | //p3 |
| int b = 1;  printf("3.%d\n", b); | //p4 |
| } printf("4.%d\n", b);  b = f (b);  printf("5.%d\n", b);  return 0; | //p5 |
| } | //p9 |

* 1. For each line marked with //p#, define the symbol table if dynamic scoping is used[[3]](#footnote-3). For example, for p1 and p2,

p1:

* + 1. :: int = 3 globalb :: int = 5 global f :: int -> int function main :: int function

p2:

a :: int = 3 global b :: int = 3 local to main :: int = 5 global f :: int -> int function main :: int function

* 1. According to the dynamic scoping, what would be the output of the program (i.e., the prints instandard output)?

1. Consider the following program.
   1. #include <stdio.h>
   2. int a = 53;
   3. int b = 120;
   4. int c = 36;
   5. int g (int a) {

|  |  |
| --- | --- |
| (6) | printf("4.%d\n", b); |
| (7) | int c = b; |
| (8) | int b = a + c; |
| (9) | printf("5.%d\n", b); |
| (10)  (11) } | return b; |

(12) int f (int a){

|  |  |
| --- | --- |
| (13) | int b = a + 5; |
| (14) | printf("3.%d\n", b); |
| (15)  (16) } | return g (b); |

(17) int main () {

|  |  |
| --- | --- |
| (18) | printf("1.%d\n", b); |
| (19) | int b = a; |
| (20) | printf("2.%d\n", b); |
| (21) | int c = f (b); |
| (22) | printf("6.%d\n", c); |
| (23)  (24) } | return 0; |

* 1. Define the symbol table if static scoping is used, after each of the following lines.

Lines: 2, 3, 4, 5, 7, 8, 11, 12, 13, 16, 17, 19, 21 and 24.

* 1. According to the static scoping what would be the output of the program (i.e., the prints in standardoutput)?

1. Consider the same program, given in the previous question. Define the symbol table if dynamic scoping is used, after each of the following lines.

Lines: 17, 19, 12, 13, 5, 7, 8, 11, 16, 21, and 24.

Note that the sequence of the line above follow the order of program execution.

1. In C, scope analysis *is* static [↑](#footnote-ref-1)
2. You can use your way of specifying labeled stacks. No need to follow the syntax used here. Note that for each name a stack should be defined, and items within each stack must be delimited from the adjacent ones. [↑](#footnote-ref-2)
3. We are just *assuming* that the scope analysis is done dynamically for C. [↑](#footnote-ref-3)