TA2

Wesley Brown

3.2

(Java) When creating a program if a local variable was static then it would be accessible from any point throughout that program. Within java this would create problems as it would cause many issues. In the bellow program there are issues because i will be changed to two causing infinite recursion.

First(){

static int i=0;

for(i>0; i--){

second();

}

return i;

}

Second(){

Static int i=2;

}

(Sheme)

(define (sum seq)

(if (null? seq)

0

(+ (car seq) (sum (cdr seq)))))

(display (sum '(5 6 1 8 3 7)))

(display "\n")

This would have issues with the seq variable as it would not provide the correct value each time.

3.4

Java:

Class Scope{

public static void main(String[]args){

for(int i=10; i>0;i--){

int x=i;

}

Sytem.out.println(x);//this is out of scope

First();

Sytem.out.println(z);//this is also out of scope

Second(x);//this is definitely out of scope

}

Public void First(){

int z=18;

}

Public void Second(int x){}

}

3.5

Prints: a=1 b=2 a=3 b=1 a=1 b=2 (C)

Prints: a=3 b=3 a=3 b=3 a=1 b=2 (modula-3)

3.7

a. In the implementation of your code you use the insert function during your while loop which mallocs memory but you never free the memory up at any point in your program causing memory leaks which end with a crash.

b. The way that you are moving variable from one array to another is correct, but it also does not allow for the data to be stored in the correct order. You do free up the memory, but you need to run reverse() on T instead of setting L = T.

3.14

Static – 1 1 2 2

Dynamic – 1 1 2 1

Why: The reason that these print differently is because when a program is statically scoped it is able to change the value stored to a variable which then has that same value for the rest of the program until it is out of scope or the value is changed again. The dynamically scoped program will only reference the values which are set/changed in the method/function which it is in, for this case the values were all changed in the set\_x() method and did not effect the print in the print\_x() method.

3.18

shallow binding only traverses up until it finds the nearest variable that it corresponds with, so for this example, set\_x and print\_x would always access the local x in foo

prints - 10203040

deep binding calls the referencing environment that the function was bound with when it was passed, so for this example, set\_x and print\_x accesses the global x when n is even and local x when n is odd

prints - 10520044

-Brandon Fung 10/29/2020