Edition 4: question 3.2, page 167.

Consider variable allocation. Give only Pascal/Java and Scheme examples. They don't need to compile/execute.

Java

```
Class{
        Main{
                Static int x = 0;
                Y += x;
        }
}
```

This would not work because local variables can not be static in java. So no matter what the program did involving the variable x would not work.

Scheme

```
(define (difference seq)
 (if (null? seq)
    0
   (- (car seq) (difference (cdr seq)))))
(display (difference '(5 6 1 8 3 7)))
(display "\n")
```

Any scheme program that has recursion would not work correctly if local variables were allocated on the stack.

Edition 4: question 3.4, page 167.

Consider live, but invisible, variables. Your examples need not compile/execute. Code:

Example 1.

```
Main {
        Int a = 1; ← local variable will be live until the end of main
        Int b = 2;
        Int c = 3;
        methodCall();← while this method is benign called a is still live but is not in scope
        because local variables are only in scope in the block they are declared in
        Int d = a + b;
        a = d + c;
}
methodCall(){
        System.out.println("Hi");
}
```

Example 2.
Private Int j = 0; ←instance variable
Static method{
… ← A instance variable will be live for the duration of a program but is not in scope static methods. So here int j is live but not in scope
}
Main{
…
}

Example three

Class 1{
Variable x ← This is a class variable. It is live throughout the whole program but only in scope in this class is it declared in.
Main{...}

}
Class 2{
… ← so variable x would be live but not in scope here
}

Edition 4: question 3.5, page 167.

Consider declaration order. Consider only the C and Modula-3 sets of rules.

C.

The program will identify static semantic errors.

print on line 11 is looking at the b declared on like 5 (which is where there is an error because a has not been declared) and the a declared on line 9

print on the line 7 is trying to print two variables that have no declaration

print on like 14 is looking at the a declared on line 2 and the b declared on line 3

Modula-3

Line 11: 1 2
Line 7: 3 3
Line 14: 1 2

Edition 4: question 3.7, page 169.

Analyze memory bugs. Figure 3.16 is within the exercise.

a. Brad is never calling free (or deleteList which then calls free). Because of this the things in memory are just always building up and never actually deleted once they are done being used.
b. This is because L and T are pointers to a node_list and just simply assigning T to L using a = dose not work in C and Brad will need to call L = insert(T→data, L);

Dynamic → 1120

In dynamic when everytime a block of code is finished it will be added to a stack presay. Then when it is looking for a variable it will look in the current block and if it is not found it will just check the blocks in the stack until it finds the variable it's looking for.

Static → 1121

When static if there is a variable declared locally it will use that if the variable is not declared locally it will look at the global varibles.

Edition 4: question 3.18, page 173.

Consider shallow and deep binding.

Shallow: 10,20,10,30
Deep binding: 10 52 33 44