1. Assembly language introduces labels, which stand for addresses within the code of the program. How does this make assembly language more convenient to write than machine language?

2. In procedural languages, variables are often associated with a memory location that stores the current value of the variable.In C, the & operator obtains the address of a variable's storage, which may be passed to other procedures. Describe how this may lead to *unsafe* memory accesses (i.e., accessing a memory location that does not exist).

3. We can simulate structured code in a language which only has labels, goto, and if...goto. For example, the pseudo-code on the right simulates the code on the left.

```
if x = 0                        if x ≠ 0 goto L1
   y := z                       y := z
end if                    L1:
x := x + y                      x := x + y
```

Rewrite the structured program given below using labels, goto, and if...goto. Assume that row, col, and matrix are previously declared.

```
row := 0
while row < ROWS:
        col := 0
        while col < COLS:
                if row = col:
                        matrix[row, col] := 1
                else:
                        matrix[row, col] := 0
                end if
                col := col + 1
        end while
        row := row + 1
end while
```

Consider the following pseudo-code:

```
procedure swap(x, y):
    var tmp = x
    x := y
    y := tmp
    print x, y
end procedure

var a = 1
var b = 2
swap(a, b)
print a, b
```

4. What does this print if parameters are passed by value?

5. What does this print if parameters are passed by reference?

6. In call-by-name, arguments are passed without evaluating them first. Instead, arguments are evaluated each time their corresponding parameters are used. Assuming call-by-name, what would the following code print?

```
var g = 0

procedure foo(x):
    g := g + 1
    print x
end procedure

foo(g * 2)
```

7. For each of the lines marked A, B, and C, indicate which variables (including parameters) are in scope, and the line on which that variable was bound.

For example, on line E, the variables in scope are x (1), a (3), b (3), and y (4).

```
1: var x
2:
3: procedure first(a, b):
4:     var y
5:     -- (E)
6:     procedure second(x):
7:             var z
8:             procedure third(w):
9:                     var a
10:                    -- (A)
11:            end procedure
12:            -- (B)
13:    end procedure
14:
15:    procedure fourth(a, b):
16:            var x
17:            -- (C)
18:    end procedure
19: end procedure
```

8. Claim: In a language which has procedures and local variables, but does not allow heap allocation or recursion, the total memory requirements of the program (including code, variable storage, and return addresses) can be determined statically. Explain why or why not.