Scope Example
What does the function `test` print if the language uses static scoping? What does it print with dynamic scoping? (otherwise assume C++ syntax and semantics, e.g. call by value).

```cpp
int n = 1; // global

print_plus_n(int x) {
    cout << x + n;
}
increment_n() {
    n = n + 2;
    print_plus_n(n);
}

test() {
    int n;
    n = 200;
    print_plus_n(7);

    n = 50;
    increment_n();
    cout << n;
}
```

With Static Scoping:

8 6 50

With Dynamic Scoping:

207 104 52
Functional programming Questions:

1. a) What is a first class citizen in a programming language?  
Something that can be passed to or returned from a function and also be bound to a variable/symbol.

   b) Give an example of a first class citizen in scheme.  
   **int, and real are first class citizens in Scheme and many other programming languages. Functions are also first class citizens in Scheme.**

2. What is programming in a “purely functional style”?  
Programming without side effects, using only the composition of functions to accomplish things.

3. What is the result of the following in Scheme:

   \[(\text{map} \ (\lambda (x) \ (+ \ x \ 50)) \ (1 \ 2 \ 3 \ 4)) \rightarrow (51 \ 52 \ 53 \ 54)\]

4. Assuming that the following definitions are executed in this order:

   \[
   \begin{align*}
   \text{(define x (3 \ 28 \ 400))} \\
   \text{(define y (cons (cdr x) (6 \ 15 \ 77)))}
   \end{align*}
   \]

   What is the result of typing the following into the Scheme interpreter:

   \[
   \begin{align*}
   \text{y} & \rightarrow ??? \quad ((28 \ 400) \ 6 \ 15 \ 77) \\
   \text{(cons 'x (cdr (cdr x)))} & \rightarrow ??? \quad (x \ 400)
   \end{align*}
   \]

5. Write a recursive Scheme function, **merge_sorted** that takes two sorted lists as parameters and returns a single list that contains all of the elements of both lists in sorted order. You can assume that the two lists: both contain only integer values > 0, and are sorted from smallest to largest. The two lists may not be of the same length.

   **Example:**

   \[
   \text{(merge_sorted (4 \ 8 \ 26) (6 \ 200))} \rightarrow (4 \ 6 \ 8 \ 26 \ 200)
   \]

   \[
   \begin{align*}
   \text{(define (merge_sorted x y}} \\
   \text{(cond ((null? x) y)} \\
   \text{((null? y) x)} \\
   \text{((< (car x) (car y))} \\
   \text{ (cons (car x) (merge_sorted (cdr x) y)))} \\
   \text{(else (cons (car y) (merge_sorted (cdr y) x))))))}
   \end{align*}
   \]