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).

```
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:

```
(map (lambda (x) (+ x 50)) ‘(1 2 3 4)) =>(51 52 53 54)
```

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

```
(define x ‘(3 28 400))
(define y (cons (cdr x) ‘(6 15 77)))
```

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

```
y                =>  ???    ((28 400) 6 15 77)

(cons ‘x (cdr (cdr x)))  =>  ???    (x 400)
```

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:

```
(merge_sorted ‘(4 8 26) ‘(6 200)) => (4 6 8 26 200)
```

```
(define (merge_sorted x y)
  (cond ((null? x) y)
        ((null? y) x)
        ((< (car x) (car y))
           (cons (car x) (merge_sorted (cdr x) y)))
        (else (cons (car y ) (merge_sorted (cdr y) x)))))
```