1. (15 pts) For 5 points each, compare and contrast 3 of the 4 pairs of terms. I will grade the first 3 I see.
   A. static scope and dynamic scope
   B. lifetime and scope
   C. operator overloading and operator precedence
   D. compiler and interpreter

2. (20 pts) Give an efficient accessing formula to find the A[i] element in an array with legal indexes running from lowerbound to upperbound. Be sure to note any parts that should be calculated at array allocation time (before the index operation). In addition, draw a detailed picture of the lbvector class we built for the lbvector assignment. Include all the instance variables. Show the logical and physical addresses. The user has declared this lbvector with lbvector<int> nums(6, 12) and later accesses nums[8]. If in doubt, include more detail rather than less. Be sure to show how the user’s nums[8] refers to the appropriate location in memory.

3. (8 pts) What is a class invariant? We used one for the lbvector assignment. What did you have to do to maintain it? How did it help you in building the class?

4. (8 pts) Draw diagrams showing how dangling references and memory garbage are created. Label your drawings clearly.

5. (8 pts) What are the steps in garbage collection? Give one advantage and one disadvantage to having automatic garbage collection.

6. (8 pts) We discussed five kinds of methods for classes. One of them is destructors; ignore that. List the four other kinds.
   Suppose you are building a Basket class that has methods that include isEmpty, addTo, toString, and combine along with other methods. For each method listed, indicate which kind it is. It may be more than one kind.

7. (6 pts) C++ has three parameter passing methods (by value, by reference, and by constant reference). For each of those, state when you use it and give an example of how you write the formal parameter to indicate using that method.

8. (27 pts as shown) Consider the program at right in which parameters are passed by value. Note the locations marked "// stop here", "//1", and "//2". Please answer this question on a separate sheet of paper.
   A. (4 pts) As you are executing the program, show any output. Label it clearly.
   B. (8 pts) Show the runtime stack including activation records, variables, and parameters at "// stop here". Do not erase as you build the stack. Just lightly mark through parts that no longer exist.
   C. (12 pts) Precisely describe/list the referencing environment at both "//1" and "//2" given either static or dynamic scoping. Consider all four situations.
   D. (3 pts) Show and clearly label the dynamic chain when execution stops at "// stop here".

1 int main ( ) {
2   int x = 3;
3   int y = 5;
5   void R(int w) {
6     int x = w * 3;
7     cout << "R: x is " << x << endl;
8     if (w < 10) {
9       // stop here
10     } else {
11       F(x);
12     }
13   }
15   void B(int y) {
16     int w = 4;
17     void BB( ) {
18       int x = y + 1;
19       cout << "BB: x is " << x << endl;
20       // 1
21     }
22     x = x + 3;
23     BB( );
24   }
27   void F(int y) {
28     if (y > 10) {
29       F(y - 10);
30     } else {
31       // 2
32       R(y);
33     }
36     // main starts here
37     B(x);
38     R(x+y);
39 }