1. From Lisp text: Questions 2.2, 2.4 (page 34), 2.6 (page 38), 2.13. 2.15, 2.16 (Page 49)
   2.2: ((A) (B)), (A (B (C))), (((A) (B)) (C))
   2.4: ((BOWS ARROWS) (FLOWERS CHOCOLATES))
   2.6: () = NIL, (()) = (NIL), ((())) = ((NIL)), (() ()) = (NIL NIL), (() (())) = (NIL (NIL))
   2.13:

Step	Result
Start	(((FUN)) (IN THE) (SUN))
CAR	((FUN))
CAR	(FUN)
CAR	FUN

Step	Result
Start	(((FUN)) (IN THE) (SUN))
CDR	((IN THE) (SUN))
CAR	(IN THE)
CAR	IN

Step	Result
Start	(((FUN)) (IN THE) (SUN))
CDR	((IN THE) (SUN))
CAR	(IN THE)
CDR	(THE)
CAR	THE

Step	Result
Start	(((FUN)) (IN THE) (SUN))
CDR	((IN THE) (SUN))
CDR	((SUN))
CAR	(SUN)
CAR	SUN

   2.15: ((A B) (C D) (E F))

Function	Result
CAR	(A B)
CDDR	((E F))
CADR	(C D)
CDAR	(B)
CADAR	B
CDDAR	NIL
CAAR	A

CDADDR	(F)
CADADDR	F

2.16: An error will be given because you can't do CAR of FRED

2. From Sebesta Chapter 1 review questions: Questions 6 through 16, 20 through 25, and 29.

6. UNIX is primarily written in C. This was actually one of the reasons C became so popular - it was used to write an entire operating system that became widely used.

7. When you allow user-defined operator overloading (like being able to define what + means for your own classes), it can make code harder to read because:

- The same operator might do very different things for different types
- Readers have to look up what each operator actually does
- Someone might use operators in non-intuitive ways (like using + for subtraction)

9. One example of lack of orthogonality in C is that arrays can be returned from functions by address but cannot be passed by value. Another is that some array operations work differently when arrays are parameters versus when they're not.

10. ALGOL 68 was specifically designed with orthogonality (consistency in how features can be combined) as a key principle. Every combination of features that makes sense syntactically was meant to be valid semantically.

11. The goto statement is typically the most primitive control statement. Other control structures like while loops, for loops, and if statements can theoretically be built using goto and conditional statements. (Though using goto directly is generally discouraged in modern programming.)

12. A reliable program:

- Performs its intended function correctly
- Handles unexpected inputs and conditions gracefully
- Maintains data integrity
- Is consistent in its behavior
- Has minimal bugs and doesn't crash
- Produces the same output for the same input every time
- Properly handles error conditions

13. Why is type checking parameters of a subprogram important? Type checking parameters is crucial because it:

- Catches errors early (at compile time rather than runtime)
- Ensures data being passed matches what the subprogram expects
- Prevents data corruption from mismatched types
- Makes code more reliable and easier to debug

14.What is aliasing? Aliasing occurs when multiple variables refer to the same memory location. This can lead to unexpected behavior because:

- Changes through one variable affect all aliases
- Makes code harder to understand and debug
- Can create subtle bugs in parallel programming

15. What is exception handling? Exception handling is a programming language feature that:

- Manages runtime errors gracefully
- Separates error-handling code from normal code
- Allows programs to recover from unexpected situations
- Makes programs more robust and reliable

16. Why is readability important to writability? While not explicitly covered in the text, readability affects writability because:

- Code that's easier to read is easier to maintain
- Clear, readable code makes it easier to spot and fix bugs
- Good readability makes it easier to modify and extend code
- Teams can collaborate more effectively on readable code
- Understanding existing code is necessary for writing new code that integrates with it

20. In the 1970s, two major programming language deficiencies discovered were:

- Lack of support for abstraction and modularization
- Poor control structures that made structured programming difficult

21. The three fundamental features of object-oriented programming are:

- Encapsulation (bundling data and methods that operate on that data)
- Inheritance (ability to create new classes based on existing classes)

- Polymorphism (ability to use different object types through the same interface)

22. Simula-67 was the first language to support all three fundamental features of object-oriented programming

23.From a design perspective, some conflicting criteria include:

- Simplicity vs Expressiveness (easier to learn vs more powerful features)
- Efficiency vs Safety (faster execution vs more runtime checks)
- Flexibility vs Security (more freedom vs more restrictions)

24. The text explicitly states "The major methods of implementing programming languages are compilation, pure interpretation, and hybrid implementation."

25. The text implicitly suggests compiled code runs faster through its discussion of hybrid systems and JIT compilation being used to improve performance. It mentions that Java moved from pure interpretation to JIT compilation "for faster execution."

29. Advantages of Pure Interpretation:

1. Easier Debugging

- The interpreter can provide better error messages since it's executing code line by line
- Can stop execution at any point and examine program state
- Easier to implement interactive debugging features

2. Platform Independence

- No need to compile for different machine architectures
- Code can run anywhere the interpreter runs
- Similar to how the text mentions Java's bytecode providing portability

3. Memory Efficiency

- No need to generate and store machine code
- Only needs to load what's currently being executed
- Useful for small or memory-constrained systems

4. Development Speed

- No compilation step means faster development cycle
- Immediate feedback during development

- The text mentions this concept when discussing how "the interpreter is used to develop and debug programs"

5. Dynamic Features

- Easier to implement features like dynamic typing
- Can modify code during runtime
- Supports interactive programming environments

6. Interactive Programming

- Allows for immediate execution of commands
- Good for learning and experimentation
- Useful for interactive shell environments