Questions:

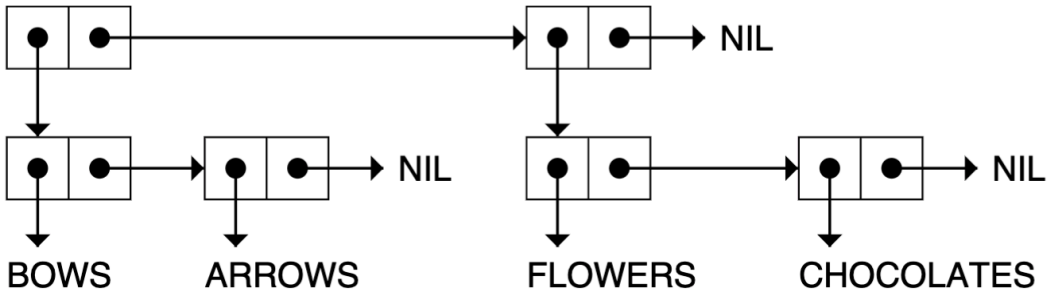
1. From Lisp text: Questions 2.2, 2.4 (page 34), 2.6 (page 38), 2.13. 2.15, 2.16 (Page 49)

2.2.
Which of these are well-formed lists?
That is, which ones have properly balanced parentheses?

(A B (C) No

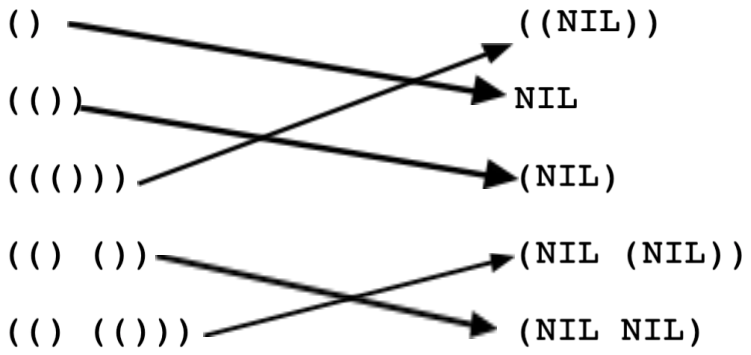
```
((A) (B))CORRECT
A B )(C D)
(A (B (C))
(A (B (C)))CORRECT
(((A) (B)) (C))CORRECT
```

2.4. What is the parenthesis notation for this cons cell structure?



The parenthesis notation for the above cons cell structure is:
((BOWS ARROWS) (FLOWERS CHOCOLATES))

2.6. Match each list on the left with a corresponding list on the right by substituting NIL for () wherever possible. Pay careful attention to levels of parenthesization.



2.13. Write down tables similar to the one above to illustrate how to get to each word in the list (((FUN)) (IN THE) (SUN)).

Step	Resutlts
Start	(((FUN)) (IN THE) (SUN))
CADDR	SUN
CDADR	THE
CAADR	IN
CAAAR	FUN

2.15. Using the list `((A B) (C D) (E F))`, fill in the missing parts of this table.

Function	Result
`CAR`	`(A B)`
`CDDR`	(E F)
`CADR`	(C D)
`CDAR`	NIL
CDAAR	`B`
`CDDAR`	NIL
CAAAR	`A`
`CDADDR`	F
CDADDR	`F`

2.16 What does CAAR do when given the input (FRED NIL)?
FRED

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

6. In what language is most of UNIX written?

Most of UNIX is written in C.

7. What is the disadvantage of having too many features in a language?

Too many features can make a language confusing and hard to learn or use. It also makes programs harder to read.

8. How can user-defined operator overloading harm the readability of a program?

It can make the code confusing if operators are used in ways people don't expect, like using + for something other than addition.

9. What is one example of a lack of orthogonality in the design of C?

Arrays and pointers in C are similar but not completely interchangeable. For example, you can't assign a new value to an array name.

10. What language used orthogonality as a primary design criterion?

The language ALGOL 68 focused on orthogonality.

11. What primitive control statement is used to build more complicated control statements in languages that lack them?

The goto statement is used to create more complex control statements.

12. What does it mean for a program to be reliable?

A program is reliable if it works correctly, even in unexpected situations, and doesn't crash.

13. Why is type checking the parameters of a subprogram important?

Type checking makes sure the data being passed matches what the program expects. This helps catch errors before running the program.

14. What is aliasing?

Aliasing happens when two or more variables point to the same spot in memory. This can cause bugs if one changes the data without the other "knowing."

15. What is exception handling?

Exception handling is a way to deal with errors in a program so it doesn't crash and can recover or handle the problem.

16. Why is readability important to writability?

If code is easy to read, it's also easier to write new parts or fix problems. You don't have to waste time figuring out what's already there.

20. What two programming language deficiencies were discovered as a result of the research in software development in the 1970s?

The two deficiencies were lack of modularity (hard to break programs into reusable components) and poor control over side effects (unexpected changes to variables caused bugs).

21. What are the three fundamental features of an object-oriented programming language?

The three features are:

Encapsulation (bundling data and methods together).

Inheritance (classes can inherit properties and methods from other classes).

Polymorphism (the ability to use the same interface for different types).

22. What language was the first to support the three fundamental features of object-oriented programming?

The first language was Simula 67.

23. What is an example of two language design criteria that are in direct conflict with each other?

Readability and efficiency often conflict. For example, highly optimized code can be harder to read.

24. What are the three general methods of implementing a programming language?

The three methods are:

Compilation (translate code into machine language).

Interpretation (execute code line by line).

Hybrid implementation (compile partway and interpret the rest).

25. Which produces faster program execution, a compiler or a pure interpreter?

A compiler produces faster execution because it translates the entire program into machine code before running it.

29. What are the advantages in implementing a language with a pure interpreter?

Implementing a language with a pure interpreter offers easier debugging, platform independence, immediate execution, better handling of dynamic features, and support for interactive environments like REPLs. It’s also simpler to develop and modify than a compiler.