Question 1:

2.2:

((A) (B)),

(A (B (C))),

(((A) (B)) (C))

Those are all correctly balanced lists

2.4:

((BOWS ARROWS)(FLOWERS CHOCOLATES))

2.6:

() = NIL

(()) = (NIL)

((())) = ((NIL))

(()()) = (NIL NIL)

(()(())) = (NIL (NIL))

2.13:

START = (((FUN)) (IN THE) (SUN))

C..AR = ((FUN))

C.AAR = (FUN)

CAAAR = FUN

START = (((FUN)) (IN THE) (SUN))

C..DR = ((IN THE)(SUN))

C.ADR = (IN THE)

CAADR = IN

START = (((FUN)) (IN THE) (SUN))

C..DR = ((IN THE)(SUN))

C.ADR = (IN THE)

CDADR = THE

START = (((FUN)) (IN THE) (SUN))

C..DR = ((IN THE)(SUN))

C.DDR = (SUN)

CDDAR = SUN

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

CAR = (A B)

CDDR = (E F)

CADR = B

CDAR = (C D)

CADR = B

CDDAR = E

CAAR = A

CDADDR = ERROR

CDDDR = F

2.16: CAAR applied to (FRED NIL)

C.AR = FRED

CAAR = ERROR

Question 2:

6: C

7: It gets really complex when many features are being used at once, and if you are trying to learn the language, there are way more things to figure out before you are able to work

8: Overloading an operator can make the readers confused because they could mean different things. For example, say we have a string 123 and we want to concatenate with another string 456 and we use + to concatenate the strings, it could be viewed as 123 + 456, which can mean addition.

9: In the expression a+b, normally a and b are added together at the same time, but if a is a pointer and b is an integer, it can cause the outcome to change because a could be pointing to a 4 byte integer, and b would have to be scaled in order to add

10: ALGOL 68

11: If would be an example, because you can build a for loop using the if statement

12: If the program performs to it's specifications under all conditions

13: It can end in errors if a type float is expected, but an int is given

14: Having two or more distinct names in a program accessing the same memory cell

15: When there are measures to catch any errors at runtime and deal with them, then continue

16: Because if a program is difficult to read, then it will be difficult to continue addition and write in it as well

20: Incompleteness of type checking and inadequacy of control statements

21: Data abstraction, inheritance, dynamic method binding

22: Smalltalk

23: Reliability and cost of execution

24: Compilation, interpretation, and hybrid

25: Compilation is faster

29: Debugging is easier due to having access to source level debugging operations