<u>Q.1</u>
2.2
(((A) (B)) (C)) is well formed list. It has properly balanced parenthesis.

2.4
The parenthesis notation for this cons cell structure is
((BOWS ARROWS) (FLOWER CHOCOLATES)).

2.6
Matching list with corresponding list
( ) ----> NIL
(( ))----> (NIL)
((( )))---->((NIL))
(( )   ( ))-----> (NIL NIL)
(( )   (( ))) ---> (NIL (NIL))

2.13
(((FUN)) (IN THE) (SUN))
CAR -((FUN))
CAR -(FUN)
CAR -FUN

CDR-(IN THE)(SUN)
CAR-(IN THE)
CAR- IN THE

CDR- (IN THE)(SUN)
CDR-(SUN)
CDR- SUN

Word	
FUN	(car (car (car ' (((FUN)) (IN THE) (SUN)))))
IN THE	(cdr (car (car ' (((FUN)) (IN THE) (SUN))))
SUN	(cdr (cdr (cdr ' (((FUN)) (IN THE) (SUN)))))

2.15

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
The operation CAAR((FRED NIL)) will cause an error because FRED is not a list, and you cannot apply the CAR function to a non-list element.

Q.2
6. Ans:
Most of UNIX is written in C, with some parts written in assembly language.

Q.16 Ans:
Readability is important to writability because:
- Clear code is easier to write. It's easier to understand.
- Readable code simplifies collaboration, debugging, and maintenance, making it quicker to write new features or fix bugs.
- Clear code improves efficiency and reduce errors, rework during writing.

Q.20 Ans:
Two programming language deficiencies that were discovered as a result of the research in software development in 1970s are:
- **Difficulty in Handling Complex Data Structures:** The programming languages were limited in how they could manage complex data structures (such as lists, trees, and graphs) efficiently.

- **Poor Support for Concurrency:** Early programming languages had limited or no built-in mechanisms for concurrency.

Q.25 Ans:
A compiler produces faster execution than a pure interpreter because it translates the entire code into machine code before running, while an interpreter translates code line-by-line during execution.

Q.3

a. (a b x d)
   (car (cdr (cdr '(a b x d))))

b. (a(b(x d)))
   (car (car (cdr (car (cdr '(a (b (x d))))))))

c. (((a (b (x) d))))
   (car (car (cdr (car (cdr (car (car '(((a (b (x) d))))))))))))

Q.4
a. (a b x d)

```
(cons 'a  (cons 'b (cons 'x ( cons 'd nil))))
```

b. (a(b(x d)))

```
(cons 'a (cons (cons 'b (cons (cons 'x (cons 'd nil)) nil)) nil))
```

c. (((a (b (x) d))))

```
(cons (cons (cons 'a (cons (cons 'b (cons (cons 'x nil) (cons 'd nil))) nil)) nil) nil)
```