CS213 : Chapter 15 (Functional Programming)

1. Introduction

- Functional languages are **based on mathematical functions**.
- Example: f(x) = x + 1

2. Mathematical Functions

- Definition: A function maps elements from a domain to a range.
- Lambda expression: Represent an anonymous **(nameless) function.**
- Example: λ(x) x * x * x → cube(x) = x^3
- (λ(x) x * x * x)(3) = 27

3. Functional Forms (Higher-order functions)

Functions that take other functions as inputs or return them as outputs.

Type	Description	Example
Composition	Combines functions: h = f ° g → h(x) = f(g(x))	$f(x)=x^3$, g(x)=x+3 → $h(x)=(x+3)^3$
Construction	Applies a list of functions to a value	→ (64,7)
Apply-to-all	Applies a function to all list elements	α(h, (3,2,4)) → (27,8,64)

4. LISP

- **First functional programming language.**
- **Data types: atoms and lists, Example: (A B (C D) E)**
- Uses lambda notation for defining functions.
- Lists are stored as linked lists.
- Example: (A B C) as function → A applied to B and C.

5. Scheme

- A simpler, modern dialect of LISP (1970s).
- **Used for teaching programming.**

Key Functions

Function	Description	Example	Result
QUOTE or '	Prevents evaluation	'(A B)	(A B)
CAR	Returns first element	(CAR '(A B C))	A
CDR	Returns rest of list	(CDR '(A B C))	(B C)
CONS	Adds element to front	(CONS 'A '(B C))	(A B C)
LIST	Creates a list	(LIST 'A 'B 'C)	(A B C)

- **Lambda**

```
((LAMBDA (L) (CAR (CAR L))) '((A B) C D))
; returns A
```

- **DEFINE (Define constants or functions)**

```
(DEFINE pi 3.14)
(DEFINE (cube x) (* x x x))
```

- **Predicates (Returns True #T OR False ())**

Function	Meaning	Example	Result
EQ?	Compare atoms	(EQ? 'A 'A)	#T
LIST?	Is it a list?	(LIST? '(A B))	#T
NULL?	Is it empty list?	(NULL? '())	#T

- **Control Flow**
  - **IF Statement**

(if condition
true-part
false-part)

  - **COND: Multiple IF cases (like switch in C++)**

(COND ((> x y) (DISPLAY "x>y"))
((< x y) (DISPLAY "y>x"))
(ELSE (DISPLAY "Equal")))

- **LET**

Defines local variables:
(LET ((x 3) (y 4)) (+ x y)) ; → 7

6. COMMON LISP

- A combination of older LISP dialects (1980s).
- **Large & complex;** includes arrays, strings, I/O, and loops.

7. ML (Meta Language)

- Static-scoped functional language (strongly typed).
- **Syntax like Pascal.**
- Performs type inference.
- Example: fun cube (x : **int**) = x * x * x;

8. Haskell

- **Purely functional** (no variables or side effects).
- Uses lazy evaluation (values computed only when needed).
- Supports list comprehensions.

9. Applications

Language	Used for
APL	Quick (throwaway) programs
LISP	Artificial Intelligence, NLP, Machine Learning
Scheme	Teaching programming

10. Comparison

Feature	Imperative	Functional
Execution	Efficient	Slower
Semantics	Complex	Simple
Syntax	Complex	Simple
Concurrency	Manual	Automatic possible