Course Code 007313 (Spring 2019)

프로그래밍언어의 개념

Concepts of Programming Language

(Lecture 03 : Chapter 2- Evolution of Major Programming Languages)

Prof. A. P. Shrestha, Ph.D.

Dept. of Computer Science and Engineering, Sejong University

Last Class

Chapter 1-Preliminaries

- Reasons for studying Concepts of PL
- Programming Domains
- Language Evaluation Criteria
- Influences in Language Design
- Language Categories

Today

Chapter 1-Preliminaries

- Implementation Methods

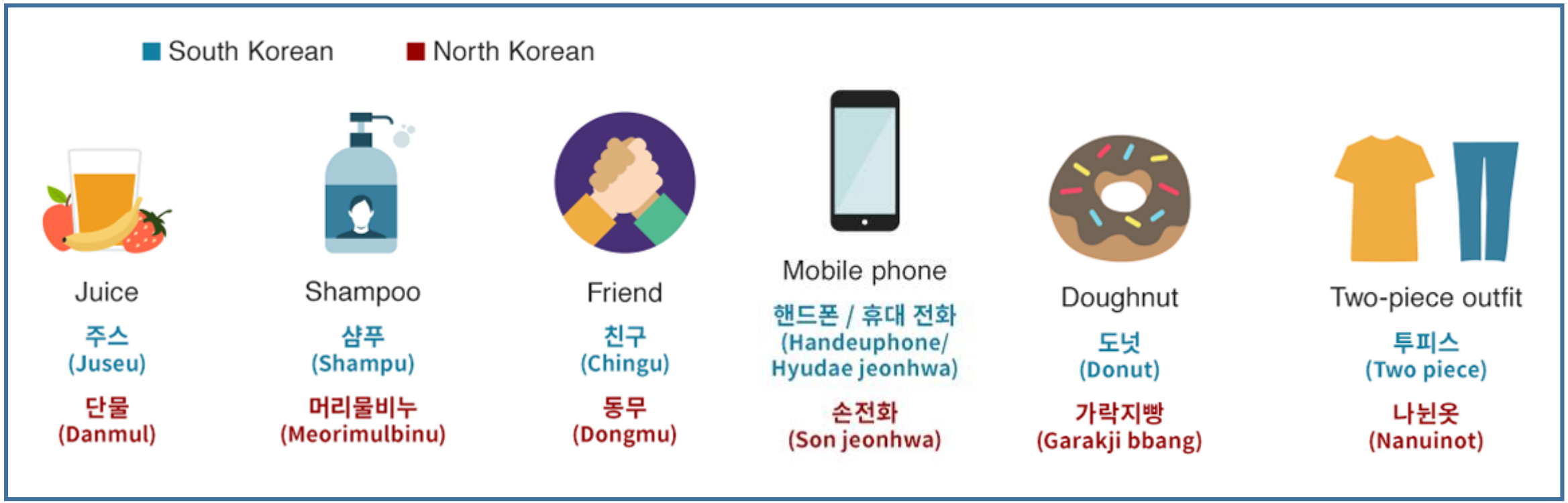
Chapter 2-Evolution of Major Programming Languages

- Language genealogy and early programming languages
- Fortran
- Lisp

Next class

- Chapter 2-Evolution of Major Programming Languages (Continue)

3.1.1 Language Evolution

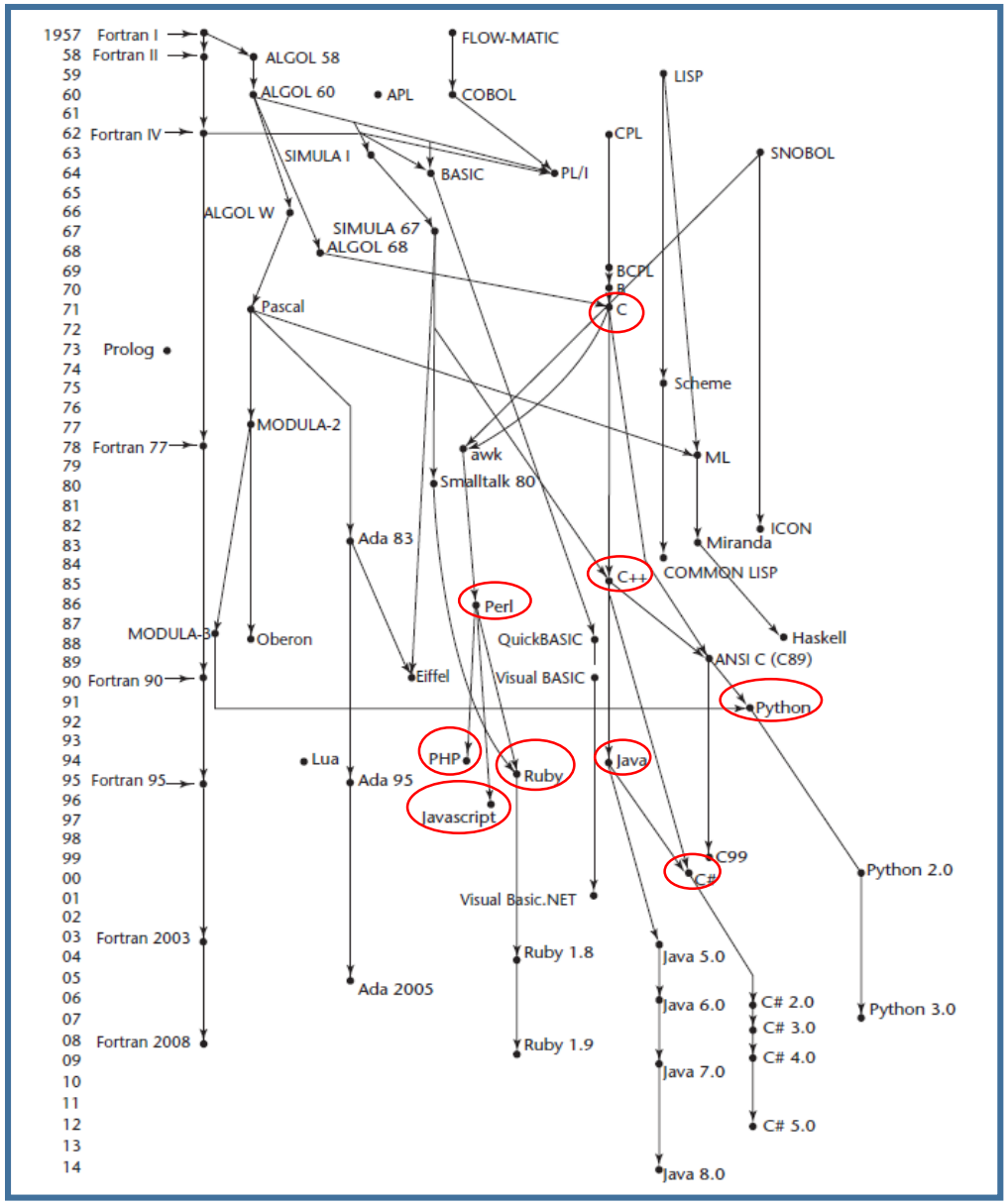


Source: https://www.bbc.com/news/world-asia-47440041

3.1.2 Genealogy of Common Languages

Genealogy:

- Successive generations of kin
- Family tree



3.2.1 Zuse's Plankalkül

- Designed in 1945, but not published until 1972
- Never implemented
- Advanced data structures
  - floating point, arrays, records
- Invariants

Fun Facts:

- Between 1936 and 1945, German scientist Konrad Zuse built a series of complex and sophisticated computers
- By early 1945, Allied bombing had destroyed all but one of his latest models the Z4
- Zuse developed a language for expressing computations for the Z4 called Plankalkül

3.2.2 Plankalkül Syntax

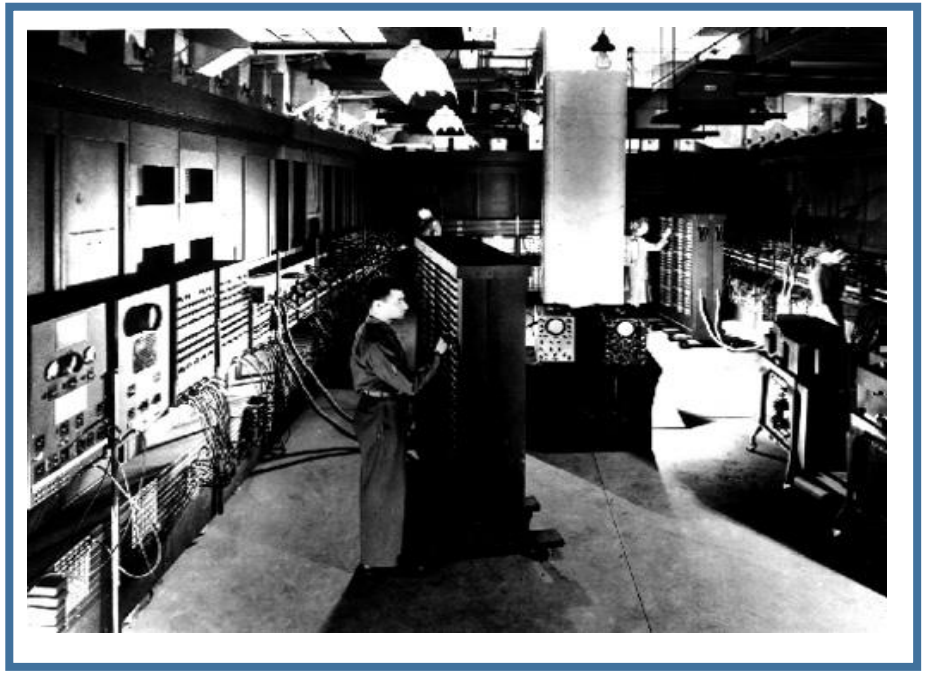
- An assignment statement to assign the expression A[4] + 1 to A[5]

	|	A + 1 => A	
V	|	4 5	(subscripts)
S	|	1.n 1.n	(data types)

- In this example, 1.n means an integer of *n* bits

3.3.1 The Dawn of Modern Computers

- Early computers (40's and early 50's) are programmed using machine code directly:
  - Limited hardware; no FP, indexing, system software
  - Computers more expensive than programmers/users
  - Poor readability, modifiability, expressiveness
  - Mimic von Neumann architecture



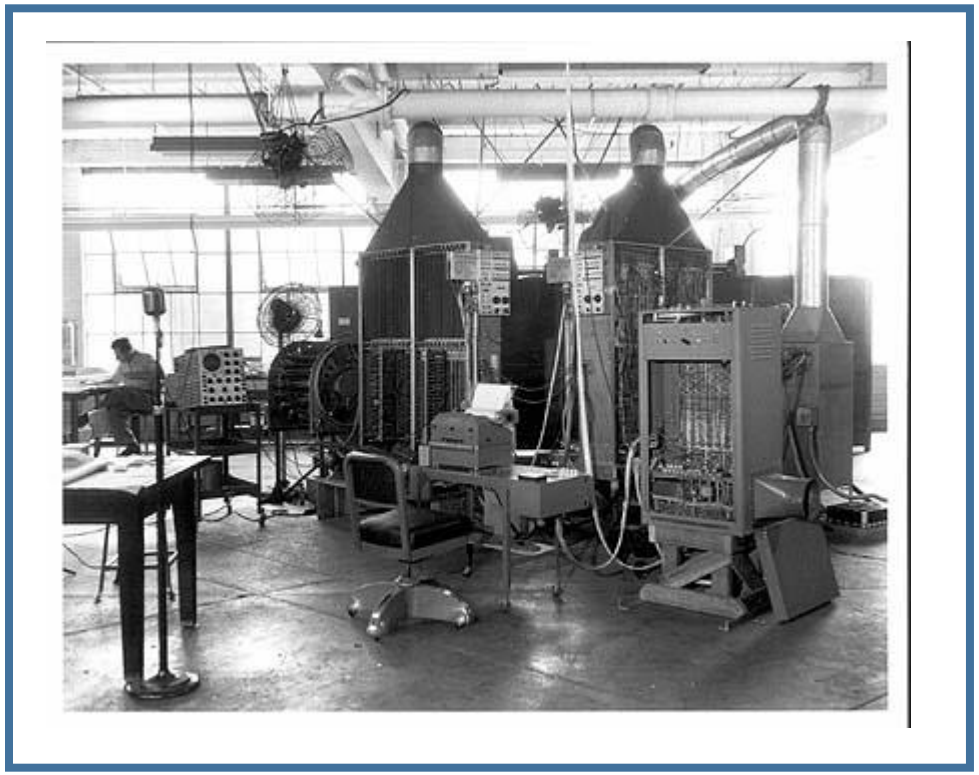
3.3.2 Early Programming-Example Short Code

- Short Code developed by Mauchly in 1949 for BINAC computers (one of the first successful stored-program electronic computers) and later used for UNIVAC I computer (first commercial electronic computer sold in the US)
  - Expressions were coded, left to right
  - Example of operations:

```
01 –        06 abs value        1n (n+2)nd power
02 )        07 +                2n (n+2)nd root
03 =        08 pause            4n  if <= n
04 /        09 (                58  print and tab
```

Statement: X0 = SQRT(ABS(Y0))
Coded in word: 00 X0 03 20 06 Y0

3.3.3 BINAC and UNIVAC



BINAC

UNIVAC

3.4.1 IBM 704 and Fortran

- First popular high-level programming language
  - Computers had small memories and were unreliable
    → machine efficiency was most important
  - Applications were scientific
    → need good array handling and counting loops

- Fortran 0: 1954
  - report titled “The IBM Mathematical FORmula TRANslating System: FORTRAN” targeting for 704 System
  - Closely tied to the IBM 704 architecture, which had index registers and floating point hardware

FORTRAN: FORmula TRANslation

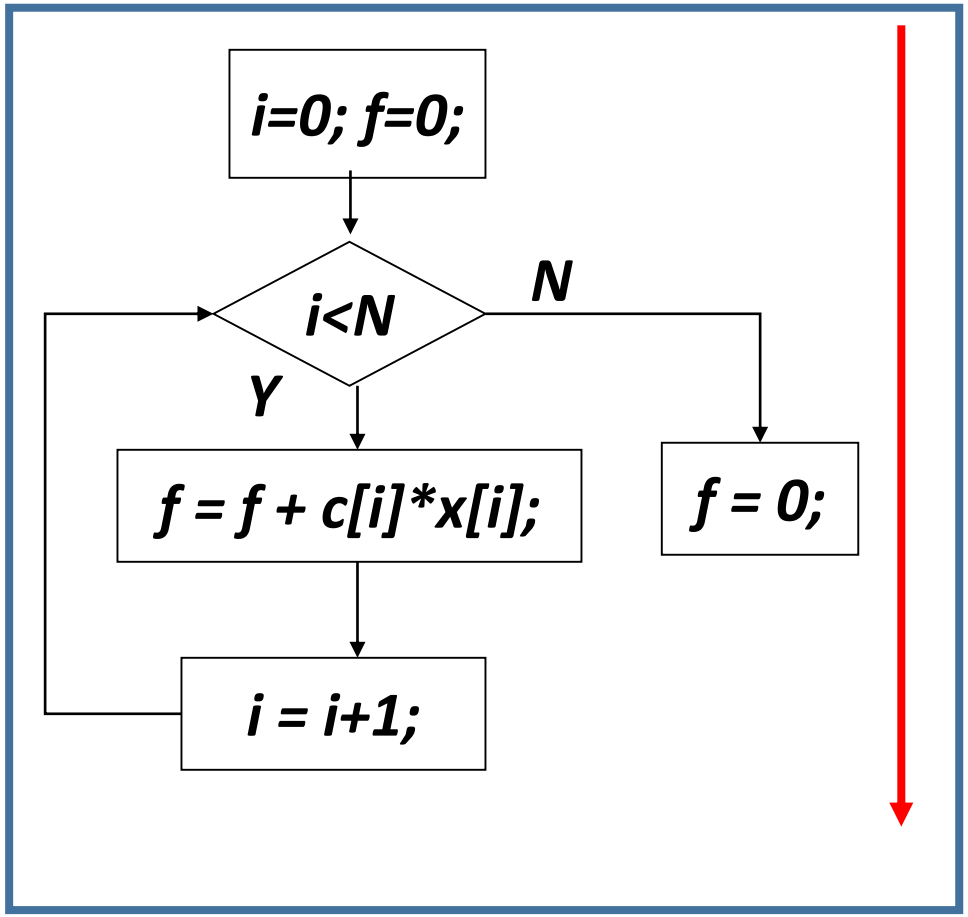
3.4.2 Fortran

- Fortran I:1957
  - Fortran 0 was modified during the implementation period and we call the implemented language as Fortran I
  - This led to the idea of compiled programming languages
  - Names could have up to six characters, formatted I/O, user-defined subroutines
  - No data typing
    - variables whose names began -> I, J, K, L, M, and N ->implicitly integer type, and
    - all others were implicitly floating-point
- Later versions evolved with more features and platform independence
  - Almost all designers learned from Fortran and Fortran team pioneered things such as scanning, parsing, register allocation, code generation, optimization

3.4.3 FORTRAN Programming Style

- Global view, top down

- Program starts from first executable statement and follow a sequential flow with go-to
  - Conceptually, a large main() including everything but without main() declaration, though FORTRAN has functions



3.4.4 A Simple FORTRAN Program

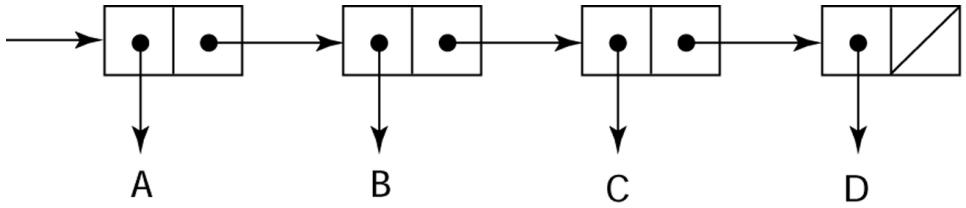
```
Program Hello

implicit none
!The implicit none statement is used to
!inhibit a very old feature of Fortran

character(40)::myname

Print *, "What is your name? "
Read *, myname
Print *, "Hello, "// myname

End Program Hello
```

3.5.1 Functional Programming: LISP

- LISP language (1958)
  - Designed at MIT by McCarthy
- AI research needed a language to
  - Process data in lists (rather than arrays)
  - Symbolic computation (rather than numeric)
- Only two data types:
  - Atoms (either symbols, which have the form of identifiers, or numeric literals)
  - Lists
- Syntax is based on *lambda calculus*

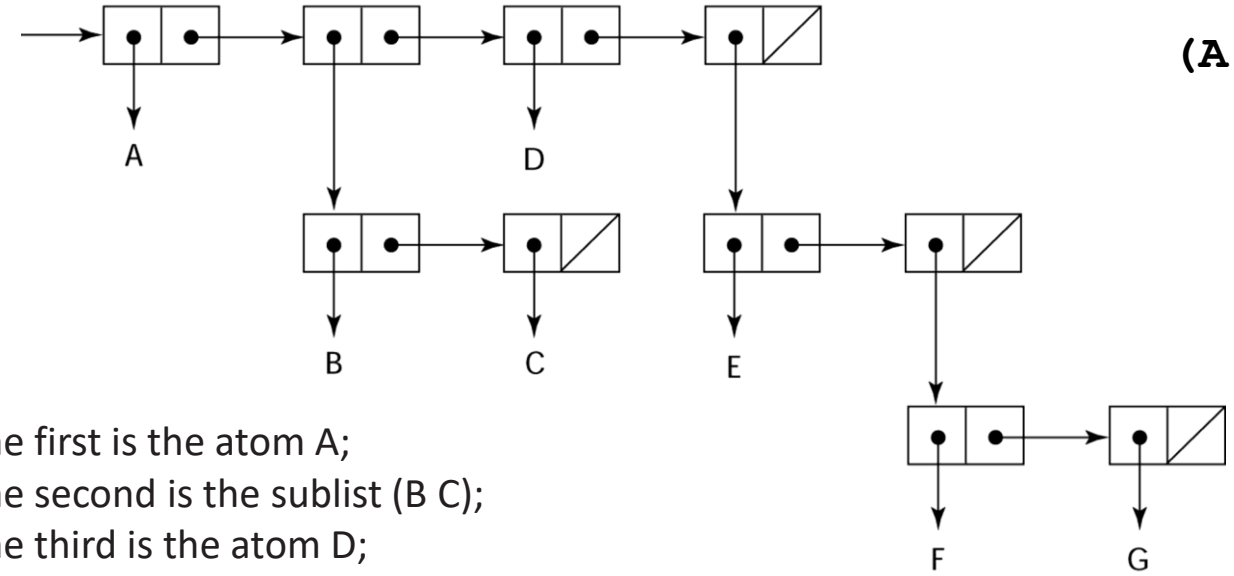
LISP: LISt Processing

3.5.2 Representation of Two LISP Lists



(A B C D)

When interpreted as data, it is a list of four elements.
When viewed as code, it is the application of the function named A to the three parameters B, C, and D.



(A (B C) D (E (F G)))

The first is the atom A;
The second is the sublist (B C);
The third is the atom D;
The fourth is the sublist (E (F G)), which has as its second element the sublist (F G).

3.5.3 LISP Features

- Pioneered functional programming
  - Computations by applying functions to parameters
  - No concept of variables (storage) or assignment
    - Single-valued variables: no assignment, not storage
  - Control via recursion and conditional expressions
    - Branches → conditional expressions
    - Iterations → recursion
  - Dynamically allocated linked lists

3.5.4 Other Functional Languages

- LISP is still the dominant language for AI
- COMMON LISP and Scheme are contemporary dialects of LISP
- ML (Meta Language), Haskell, and F# are also functional programming languages, but use very different syntax

3.5.5 Few Simple Programs in Lisp

Q) Predict the output

```
(print(* (+ 1 2) 3))
```

Note prefix notation

```
(defun sub (a b)
   (- a b))

(print (sub 15 6))
```

Note how function is defined

```
(defun factorial (n)
   (if (= n 0)
       1 (* n (factorial(- n 1)))))

(print (factorial 4))
```

Note how recursion is used

Q & A