Tutorial 1

COBOL Programming

Assignment 1 Preview

Background

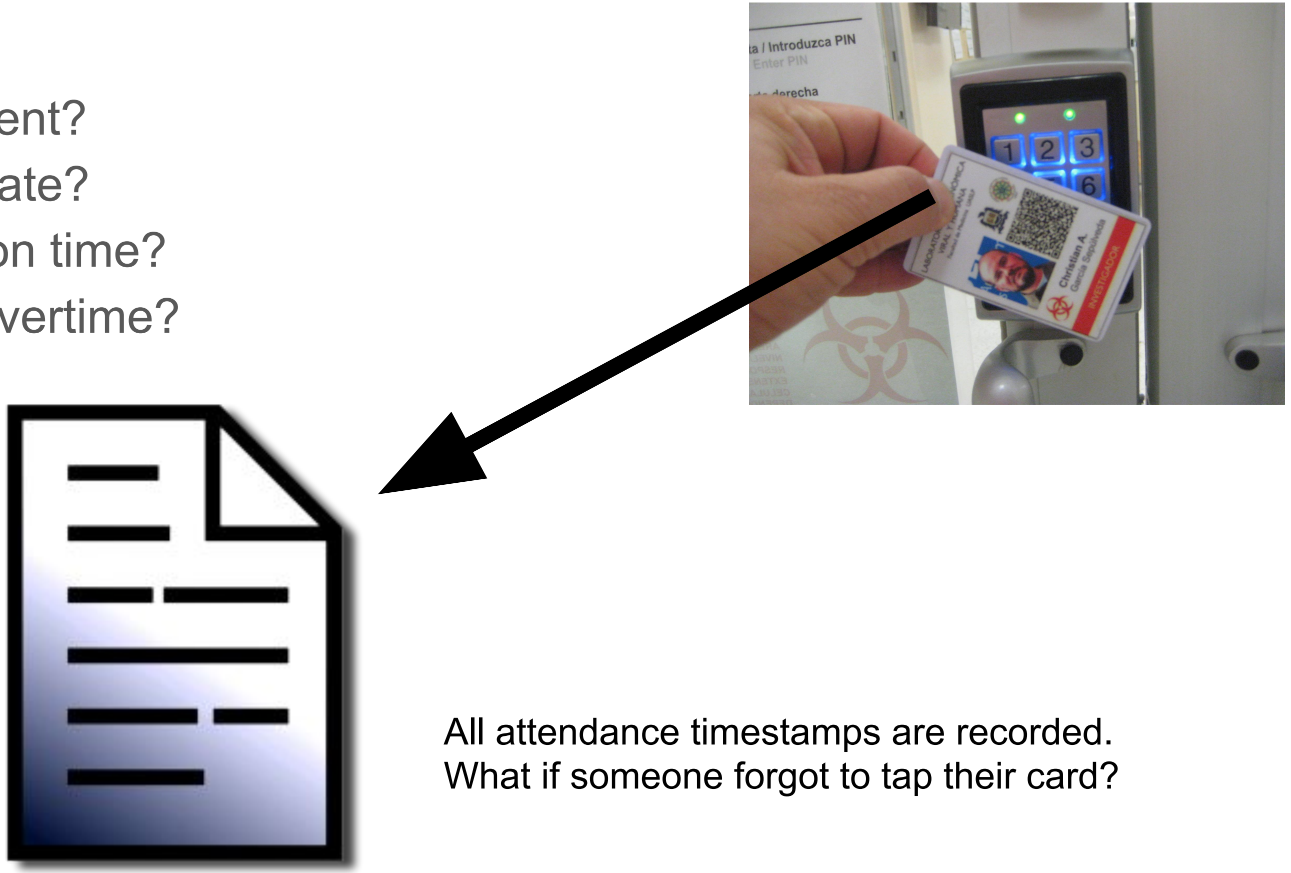
PPL corporation's employees

- work from 10am to 5pm
- tap their card when they arrive at work and depart after work

We want an attendance tracking module.

Background

- Be absent?
- Come late?
- Arrive on time?
- Work overtime?



All attendance timestamps are recorded.
What if someone forgot to tap their card?

Background

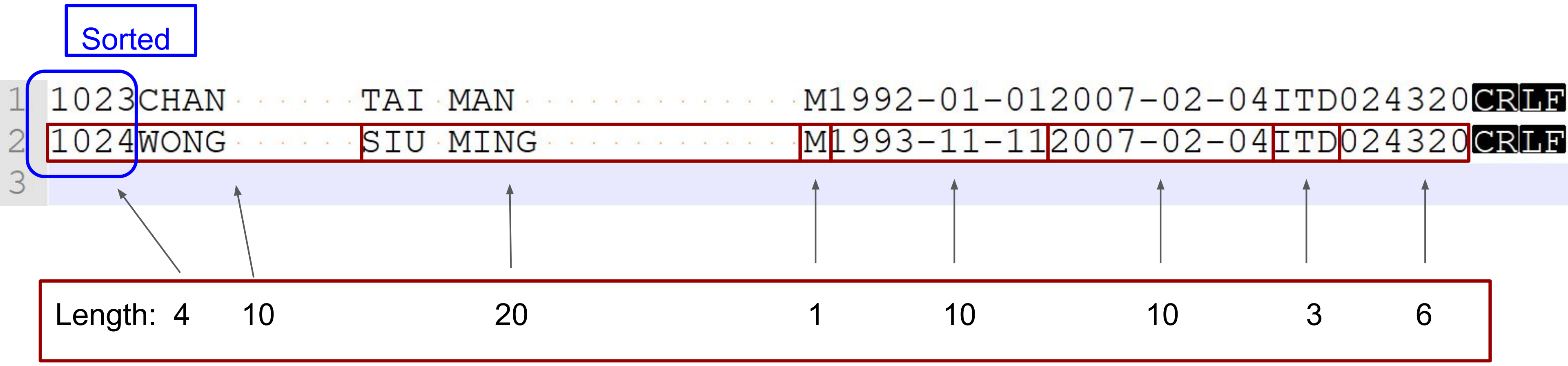
Our Goal:

- Update monthly attendance summary (monthly-attendance.txt)
  - Number of days absent
  - Number of complete 15-minute periods being late
  - Number of complete overtime work hours

- Generate a daily summary report (summary.txt)
  - Who was late?
  - Who did not come to work?
  - Who did not tap their card?

Input files

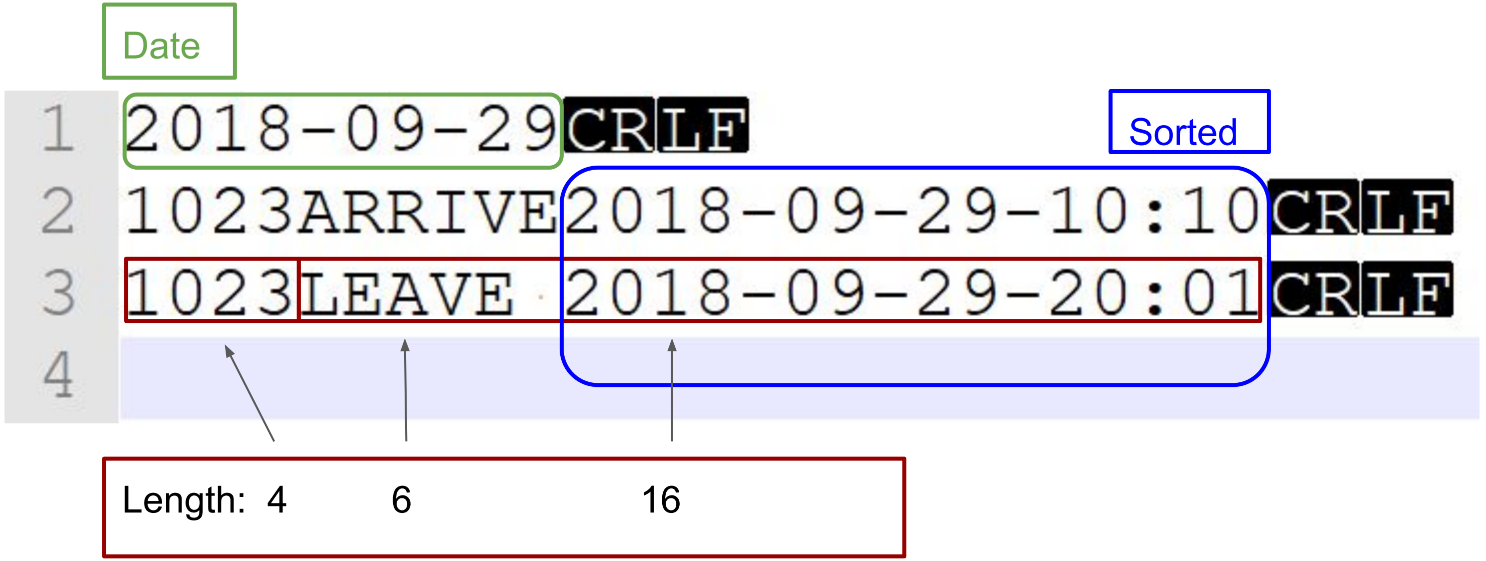
- employees.txt: a file containing all employees' information
- Example:



```
1023CHAN      TAI MAN             M1992-01-012007-02-04ITD024320CRLF
1024WONG      SIU MING            M1993-11-112007-02-04ITD024320CRLF
```

Input files

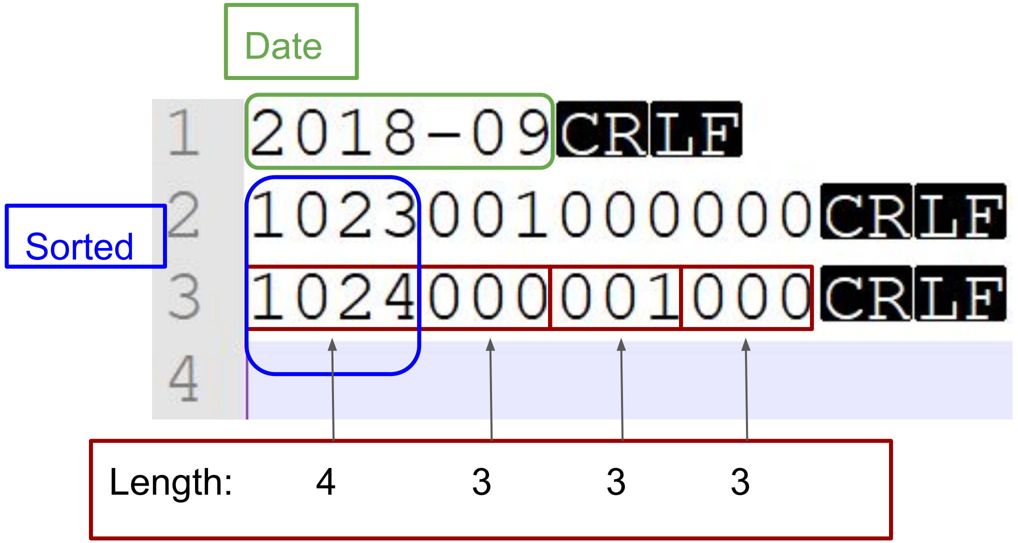
- attendance.txt: a file containing all attendance timestamps
- Example



```
2018-09-29CRLF
1023ARRIVE2018-09-29-10:10CRLF
1023LEAVE 2018-09-29-20:01CRLF
```

Input and output file

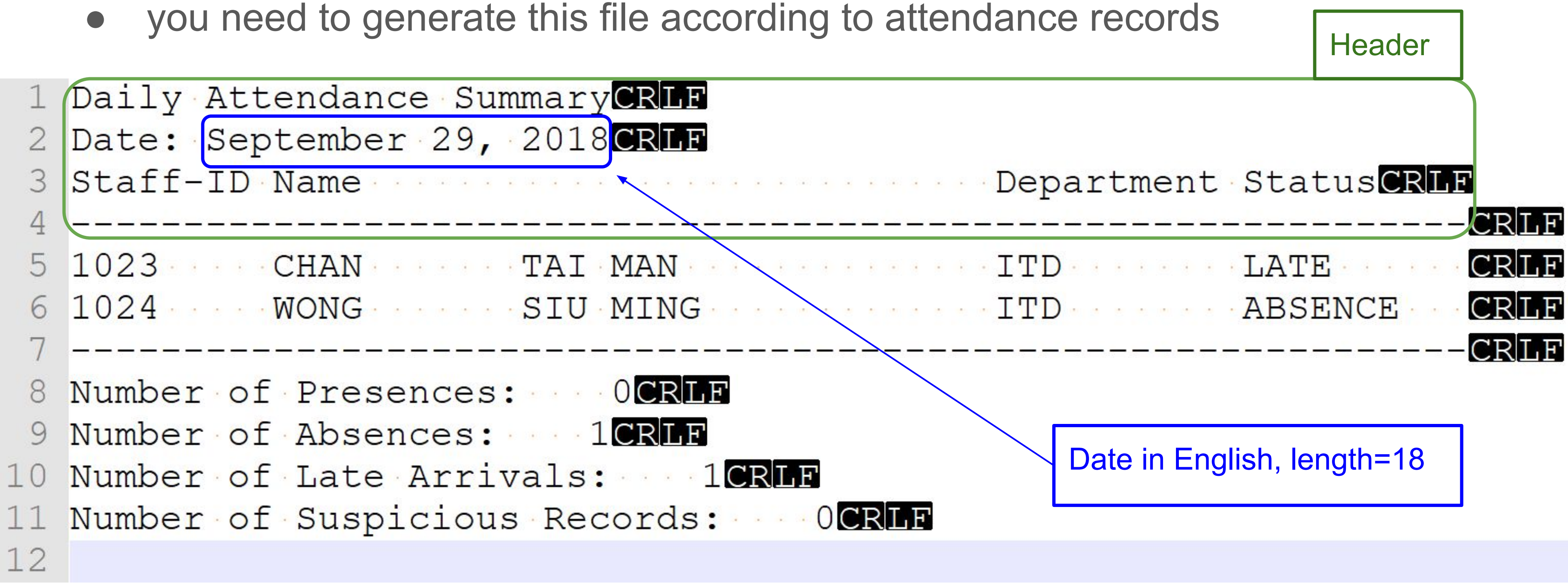
- monthly-attendance.txt: a file containing all monthly attendance information
- you need to update this file according to attendance records
- Example:



```
1 2018-09CRLF
2 1023001000000CRLF
3 1024000001000CRLF
4
```

Output files

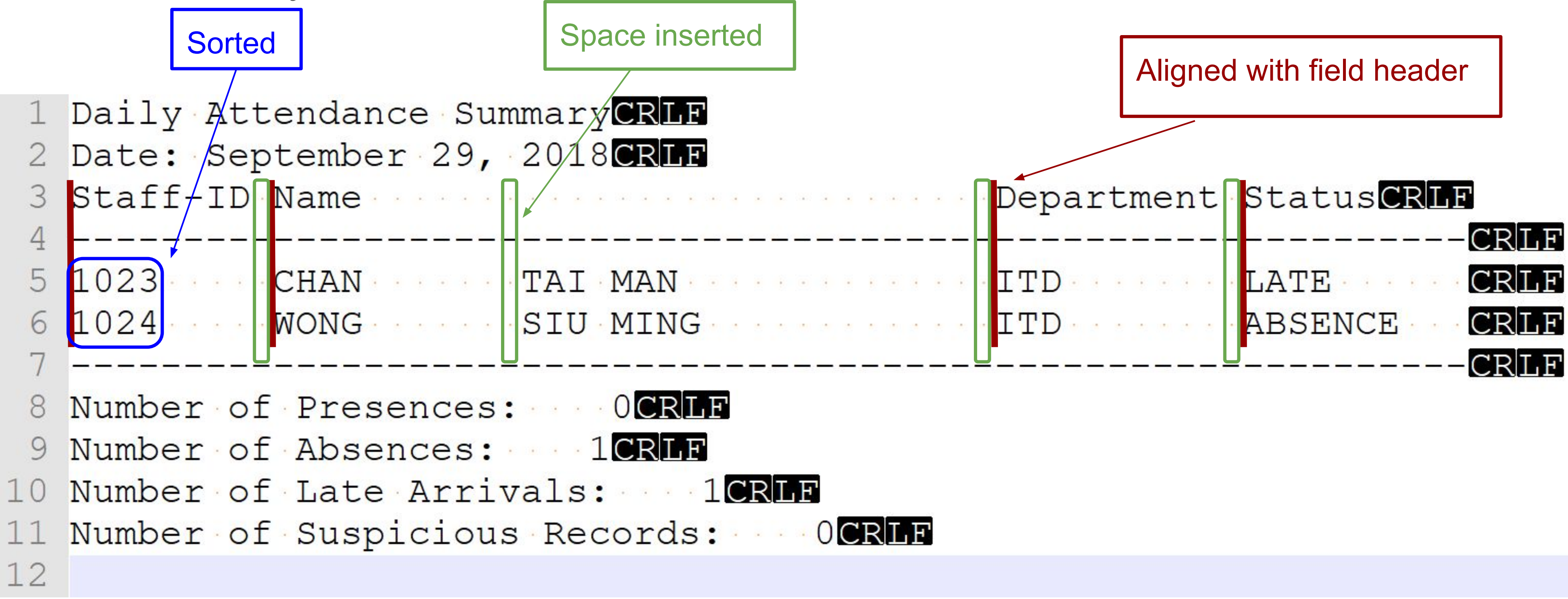
- summary.txt: a file containing attendance summary of previous day
- you need to generate this file according to attendance records



```
Daily Attendance SummaryCRLF
Date: September 29, 2018CRLF
Staff-ID Name                           Department Status CRLF
--------------------------------------------------------------CRLF
1023     CHAN       TAI MAN              ITD        LATE      CRLF
1024     WONG       SIU MING             ITD        ABSENCE   CRLF
--------------------------------------------------------------CRLF
Number of Presences:    0CRLF
Number of Absences:    1CRLF
Number of Late Arrivals:    1CRLF
Number of Suspicious Records:    0CRLF
```

CSCI 3180 Principles of Programming Languages Week 1 Tutorial

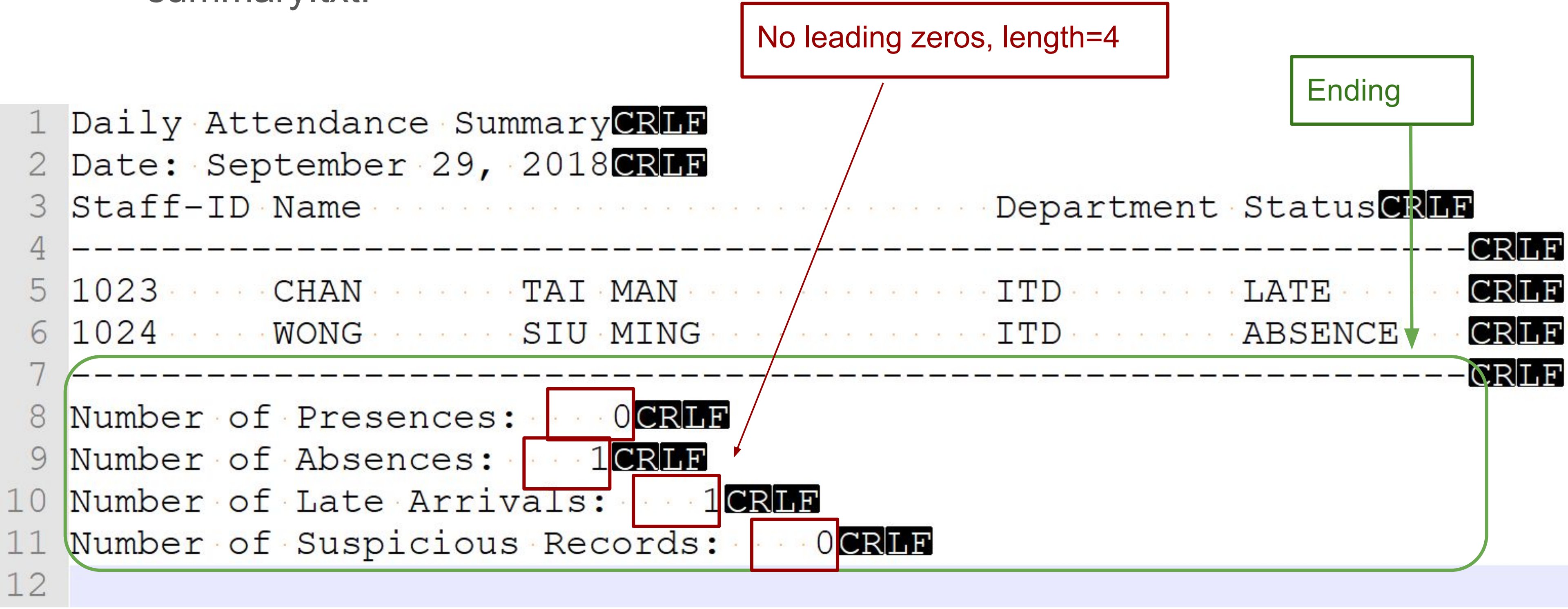
- summary.txt:



```
Daily Attendance Summary
Date: September 29, 2018
Staff-ID Name                            Department Status
--------------------------------------------------------------
1023     CHAN        TAI MAN             ITD        LATE      
1024     WONG        SIU MING            ITD        ABSENCE   
--------------------------------------------------------------
Number of Presences:    0
Number of Absences:    1
Number of Late Arrivals:    1
Number of Suspicious Records:    0
```

CSCI 3180 Principles of Programming Languages Week 1 Tutorial

- summary.txt:



```
Daily Attendance SummaryCRLF
Date: September 29, 2018CRLF
Staff-ID Name                            Department StatusCRLF
--------------------------------------------------------------CRLF
1023     CHAN       TAI MAN              ITD        LATE      CRLF
1024     WONG       SIU MING             ITD        ABSENCE   CRLF
--------------------------------------------------------------CRLF
Number of Presences:    0CRLF
Number of Absences:    1CRLF
Number of Late Arrivals:    1CRLF
Number of Suspicious Records:    0CRLF
```

COBOL

Overview

COBOL - COmmon Business Oriented Language

- One of the earliest programming languages
- Good at batch processing
- Good at file handling

You will experience ...

- the importance of data formatting
- the super English-like language

Installation

We will use GnuCOBOL v1.1 in the assignment.

- Windows
  - Download from here
  - Run set_env.bat before compilation
  - Run cobx -c test.cob to compile
- Mac
  - brew install gnu-cobol
  - The version is 2.2. Make sure you can run in Windows. (Check version: cobc -V)

Installation

GnuCOBOL v1.1 is ready in SHB924/904.

We prepare a batch file for you to set up the compilation environment.

lab904.bat

```
set lastdir=%CD%
S:
cd OpenCOBOL
CALL set_env.bat
cobc -x %lastdir%\%1
cd /d %lastdir%
```

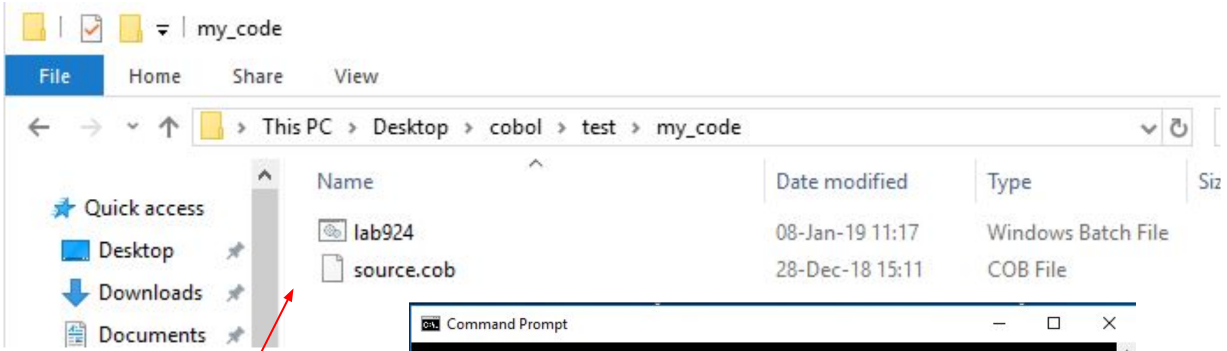
lab924.bat

```
set lastdir=%CD%
S:
cd OpenCOBOL
CALL set_env.bat
cd /d %lastdir%
cobc -x %1
```

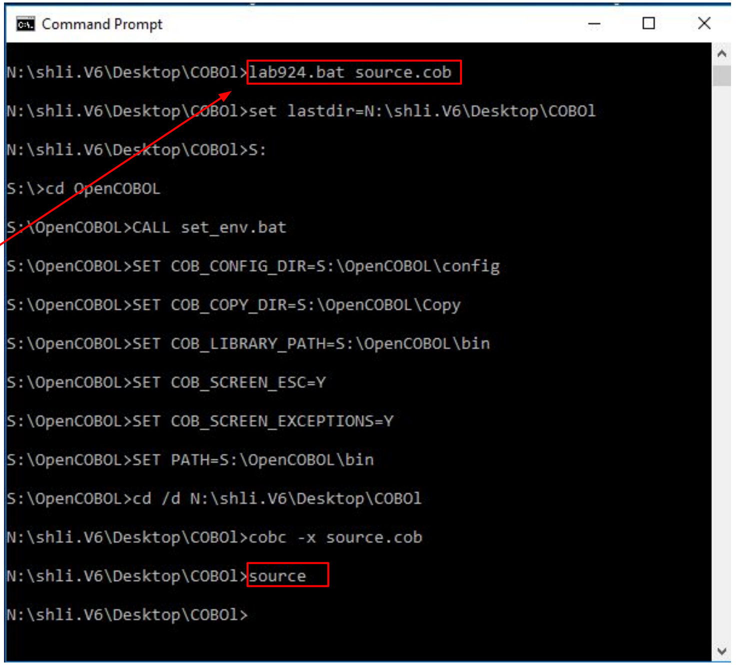
Installation

In SHB 924/904,

- Download/Copy the batch file from course homepage
- Put the batch file in the directory of your source code
- "lab924.bat source.cob" or "lab904.bat source.cob"



Name	Date modified	Type
lab924	08-Jan-19 11:17	Windows Batch File
source.cob	28-Dec-18 15:11	COB File



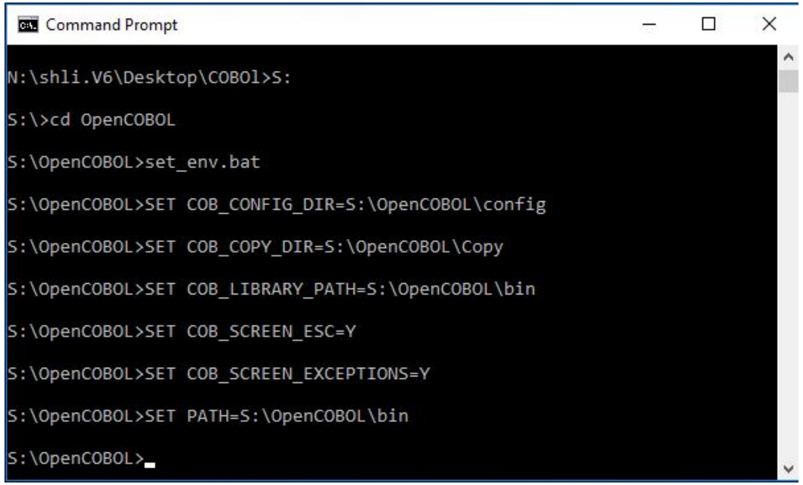
```
N:\shli.V6\Desktop\COBOl>lab924.bat source.cob

N:\shli.V6\Desktop\COBOl>set lastdir=N:\shli.V6\Desktop\COBOl

N:\shli.V6\Desktop\COBOl>S:

S:\>cd OpenCOBOL

S:\OpenCOBOL>CALL set_env.bat

S:\OpenCOBOL>SET COB_CONFIG_DIR=S:\OpenCOBOL\config

S:\OpenCOBOL>SET COB_COPY_DIR=S:\OpenCOBOL\Copy

S:\OpenCOBOL>SET COB_LIBRARY_PATH=S:\OpenCOBOL\bin

S:\OpenCOBOL>SET COB_SCREEN_ESC=Y

S:\OpenCOBOL>SET COB_SCREEN_EXCEPTIONS=Y

S:\OpenCOBOL>SET PATH=S:\OpenCOBOL\bin

S:\OpenCOBOL>cd /d N:\shli.V6\Desktop\COBOl

N:\shli.V6\Desktop\COBOl>cobc -x source.cob

N:\shli.V6\Desktop\COBOl>source

N:\shli.V6\Desktop\COBOl>
```

Installation

In SHB904/924, if the batch file fails,

- go to S:\OpenCOBOL
- run set_env.bat before compilation

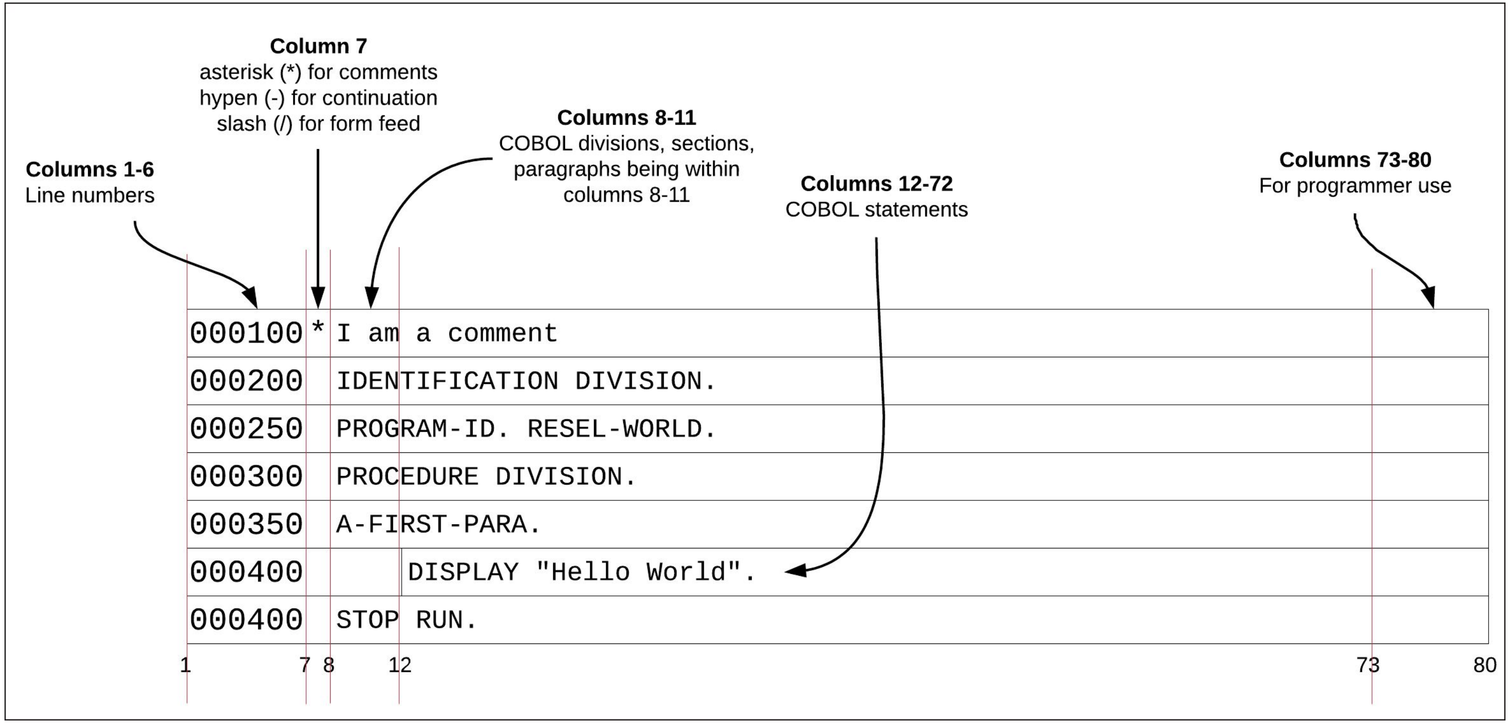
Useful window command: cd , dir



```
N:\shli.V6\Desktop\COBOl>S:

S:\>cd OpenCOBOL

S:\OpenCOBOL>set_env.bat

S:\OpenCOBOL>SET COB_CONFIG_DIR=S:\OpenCOBOL\config

S:\OpenCOBOL>SET COB_COPY_DIR=S:\OpenCOBOL\Copy

S:\OpenCOBOL>SET COB_LIBRARY_PATH=S:\OpenCOBOL\bin

S:\OpenCOBOL>SET COB_SCREEN_ESC=Y

S:\OpenCOBOL>SET COB_SCREEN_EXCEPTIONS=Y

S:\OpenCOBOL>SET PATH=S:\OpenCOBOL\bin

S:\OpenCOBOL>_
```

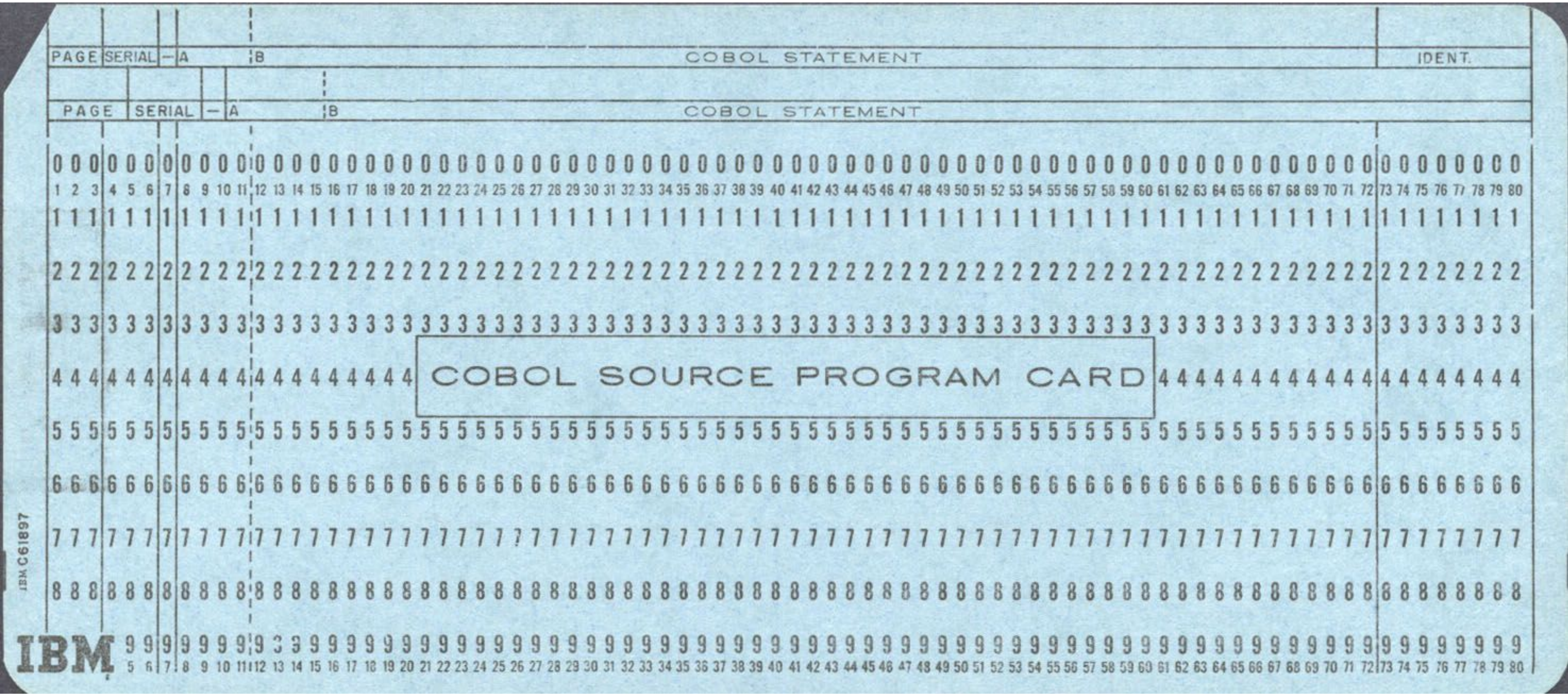
Statement format

Format is fixed.



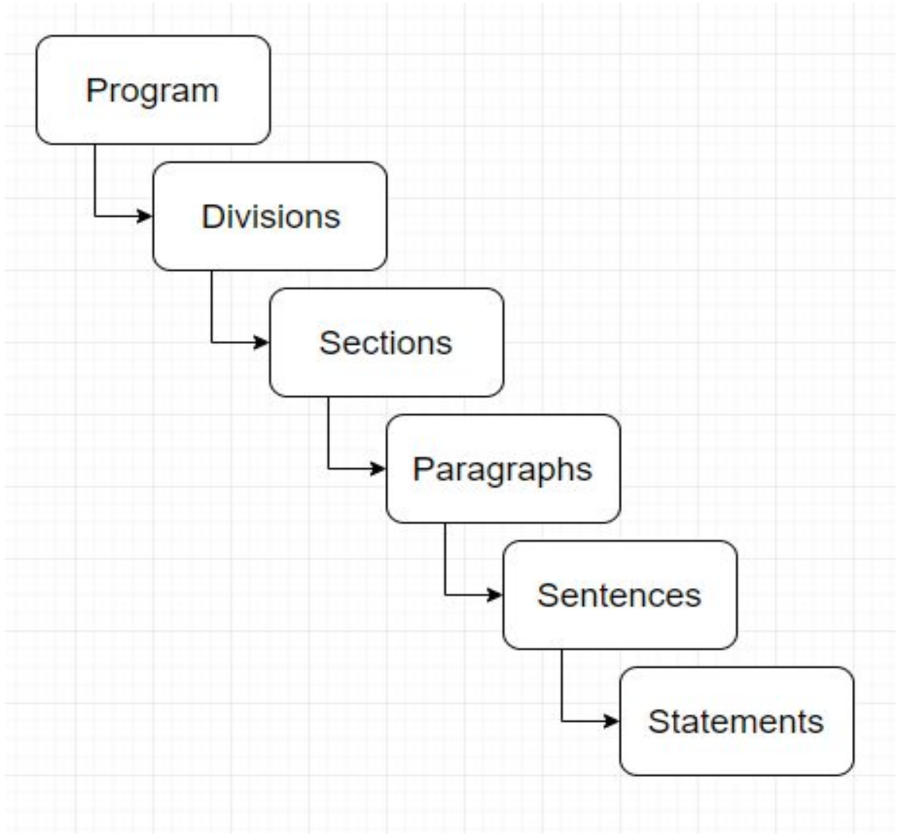
```
000100*I am a comment
000200 IDENTIFICATION DIVISION.
000250 PROGRAM-ID. RESEL-WORLD.
000300 PROCEDURE DIVISION.
000350 A-FIRST-PARA.
000400     DISPLAY "Hello World".
000400 STOP RUN.
```

CSCI 3180 Principles of Programming Languages Week 1 Tutorial



Program structure

- Statement
  - a meaningful instruction
  - start at column 12
- Sentence
  - group of statements
  - mark with period "." at the end
  - start at column 12
- Paragraph
  - block of sentences, like a function
  - start at column 8
- Divisions, sections
  - fixed
  - start at column 8



Divisions and Sections

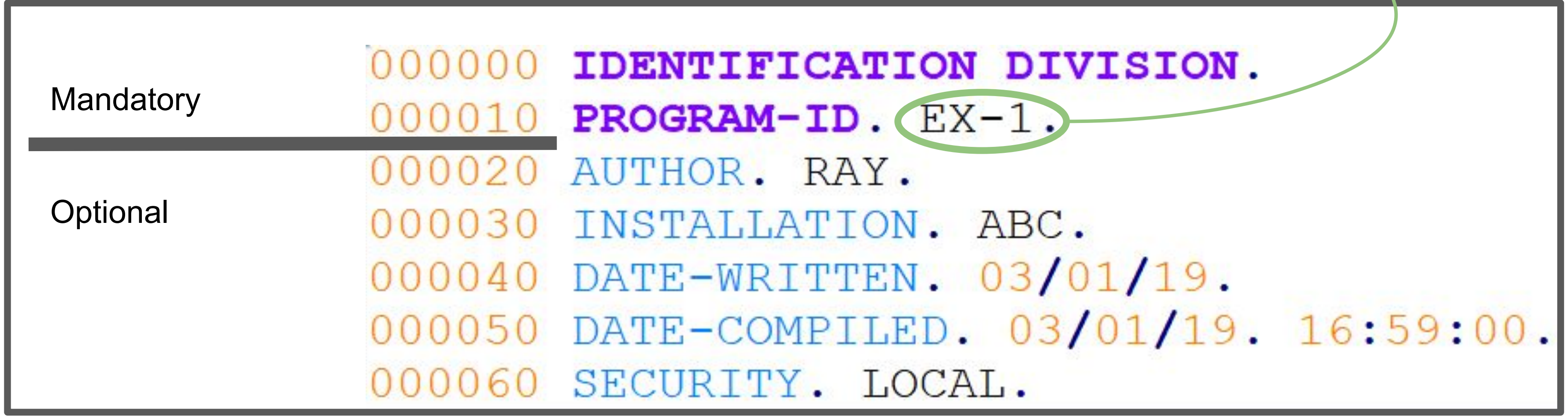
There are four divisions.

1. Identification Division (required)
2. Environment Division (optional)
    - Configuration Section
    - Input-Output Section
3. Data Division (optional)
    - File Section
    - Working Storage Section
    - Linkage Section
4. Procedural Division (optional)

Identification Division

Program name and various information.

Example:

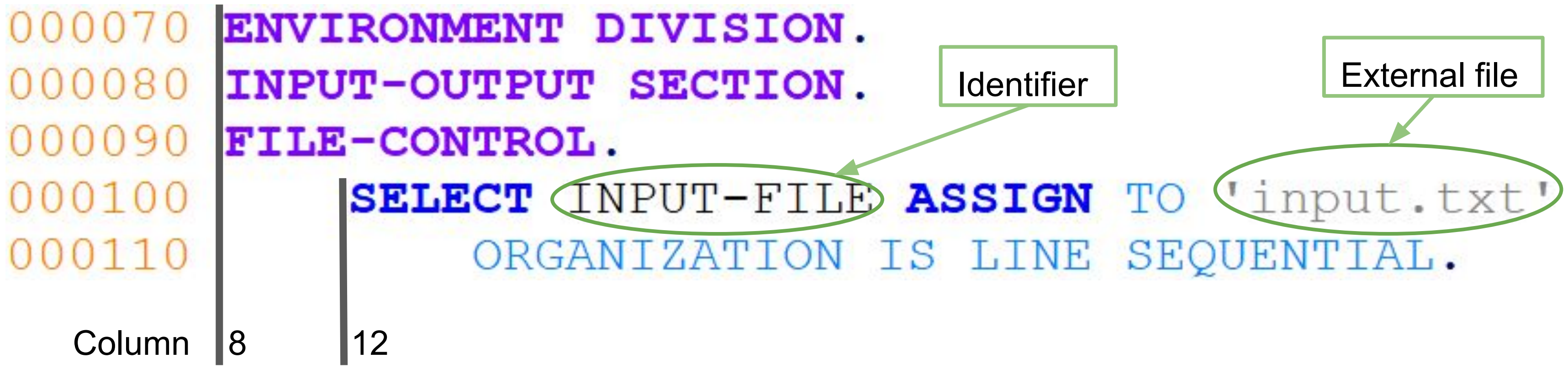


```
000000 IDENTIFICATION DIVISION.
000010 PROGRAM-ID. EX-1.
000020 AUTHOR. RAY.
000030 INSTALLATION. ABC.
000040 DATE-WRITTEN. 03/01/19.
000050 DATE-COMPILED. 03/01/19. 16:59:00.
000060 SECURITY. LOCAL.
```

Environment Division

- Configuration Section
  - Describe the hardware requirement to compile or run the program
- Input-Output Section
  - Link the identifiers from the program to external files

Example:



```
000070 ENVIRONMENT DIVISION.
000080 INPUT-OUTPUT SECTION.
000090 FILE-CONTROL.
000100     SELECT INPUT-FILE ASSIGN TO 'input.txt'
000110         ORGANIZATION IS LINE SEQUENTIAL.
```

Environment Division - Input-Output Section

Assign identifiers to files/printers

Syntax:

Defined in Data division

```
SELECT [identifier] ASSIGN TO [external file]
    ORGANIZATION IS LINE SEQUENTIAL.
```

Only mark period at the end

Can have more clauses

Optional

Data Division

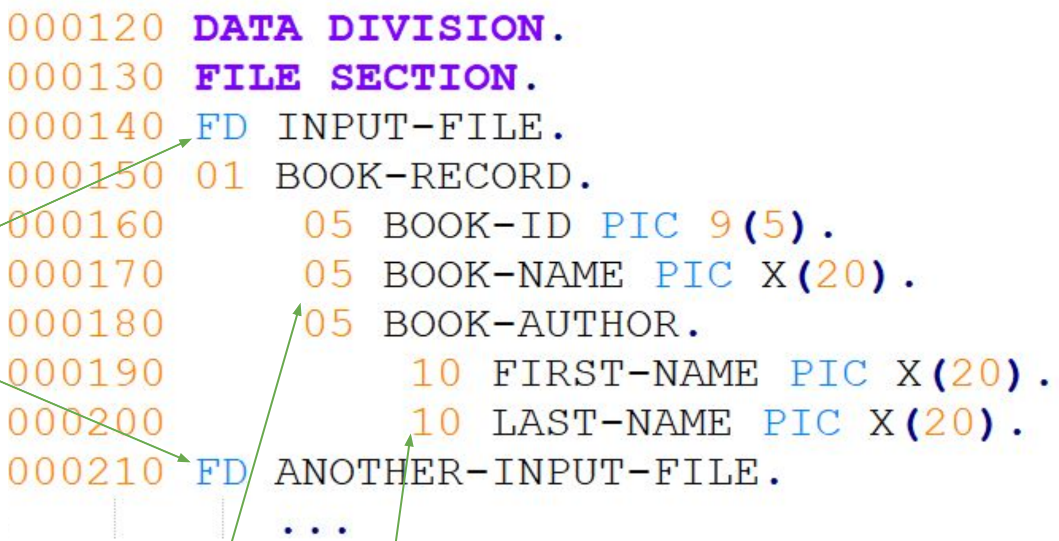
Define all data that the program will use later on, including their names, lengths, formats.

- File Section
  - Describe the data in a file, by specifying the fields within records of a file
- Working-Storage Section
  - Describe all global variables
- Linkage Section (do not use in the assignment)
  - Describe all variables from another programs

Data Division - File Section

Files consist of records.

Example:



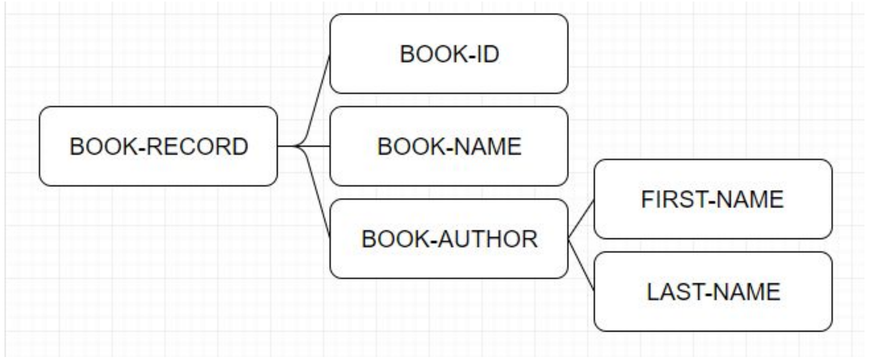
```
000120 DATA DIVISION.
000130 FILE SECTION.
000140 FD INPUT-FILE.
000150 01 BOOK-RECORD.
000160     05 BOOK-ID PIC 9(5).
000170     05 BOOK-NAME PIC X(20).
000180     05 BOOK-AUTHOR.
000190         10 FIRST-NAME PIC X(20).
000200         10 LAST-NAME PIC X(20).
000210 FD ANOTHER-INPUT-FILE.
           ...
```

File descriptions and records start at column 8.

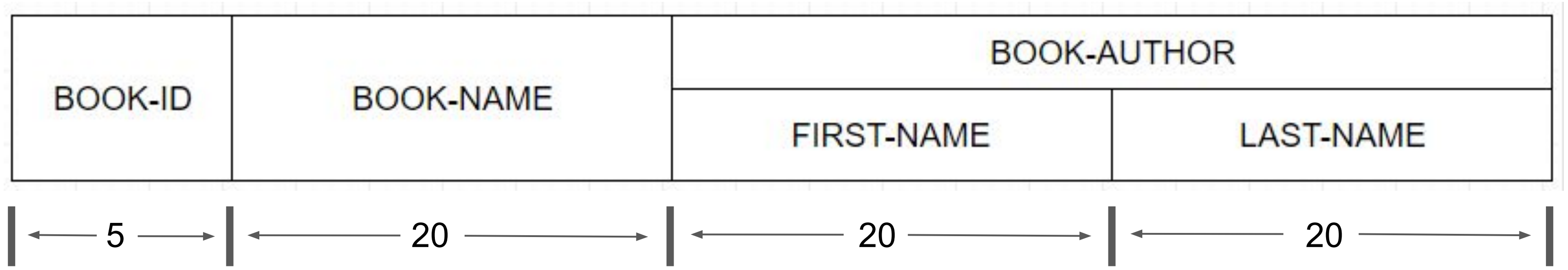
Field names start at column 12.

Data Division - File Section

We define a book record in this hierarchy.

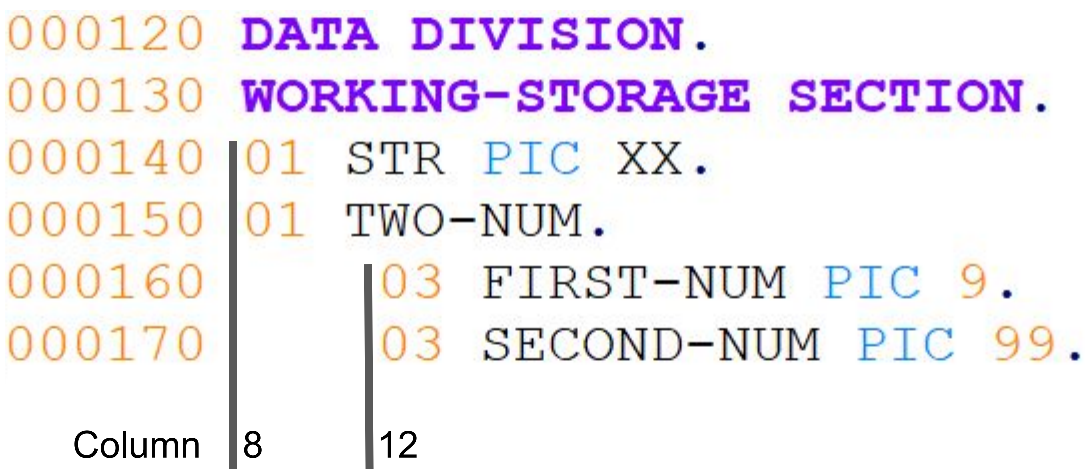


In the input file, each line represents,



Data Division - Working-Storage Section

- Define all variables to be used in the program.
- Start at column 8 for variables with level numbers 01, 77
- Start at column 12 for variables with other level numbers
- Use larger numbers for deeper levels
- Example



```
000120 DATA DIVISION.
000130 WORKING-STORAGE SECTION.
000140 01  STR PIC XX.
000150 01  TWO-NUM.
000160         03  FIRST-NUM PIC 9.
000170         03  SECOND-NUM PIC 99.
```

Variable declaration

Syntax : [level number] [name] PIC [format] (VALUE [literal])

- Level number describes the hierarchy of data.
  - From 01-49. 66, 77, 88 are reserved.
  - Hierarchy starts at small number, i.e. 01.
  - Larger number represents deeper level.
- Length of name is at most 30.
  - Alphanumeric, hyphen
- PIC clause with format describes the type and length of data.
- VALUE clause initializes the data.

Variable declaration

- Alphabet
  - Use ‘A’ in the format
- Alphanumeric
  - Use ‘X’ in the format
- Number
  - Use ‘9’ in the format
  - Max length is 18

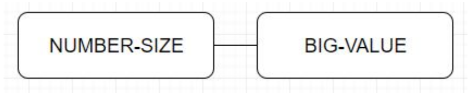
- Example:
  - 01 STR PIC XXX.
  - 01 STR PIC X(3).
  - 01 NUM PIC 9999.
  - 01 NUM PIC 9(4).
  - 01 NUM PIC 99.99.
  - 01 NUM PIC 99V99.

  (‘V’ represents decimal point.)

Variable declaration

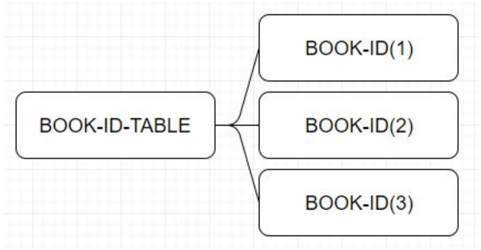
- Boolean

```
01 NUMBER-SIZE.
        88 BIG-VALUE VALUE ‘Y’.
```



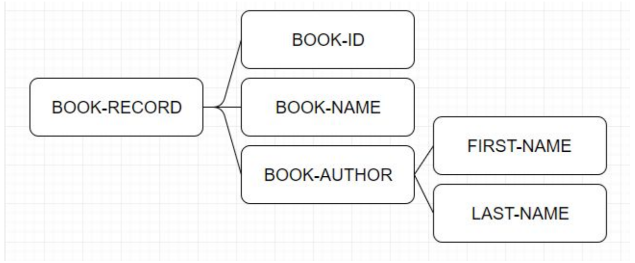
- Table

```
01 BOOK-ID-TABLE.
        05 BOOK-ID PIC 9(5) OCCUR 3 TIMES.
```



- Record

```
01 BOOK-RECORD.
        05 BOOK-ID PIC 9(5).
        05 BOOK-NAME PIC X(20).
        05 BOOK-AUTHOR.
                10 FIRST-NAME PIC X(20).
                10 LAST-NAME PIC X(20).
```

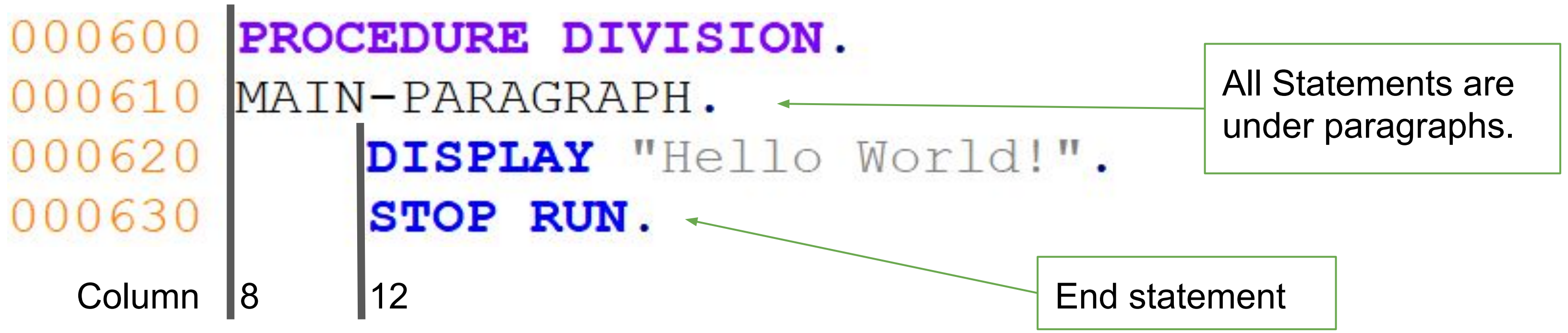


Procedural Division

Describes all operations to be performed, i.e. all your program logic

Contains multiple paragraphs and each paragraph serves as a small procedure.

Example:



```
000600 PROCEDURE DIVISION.
000610 MAIN-PARAGRAPH.
000620     DISPLAY "Hello World!".
000630     STOP RUN.
```

I/O Statement

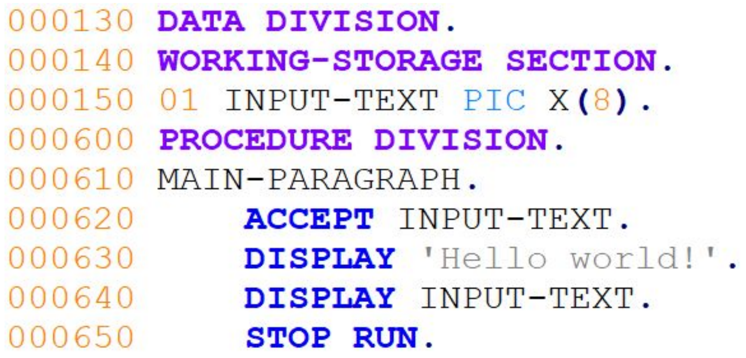
DISPLAY : output a line to the command prompt. ‘\n’ is appended.

Accept : receive data from the outside of the program

Syntax: DISPLAY [identifier] / ’......’

ACCEPT [identifier]

Example :



```
000130 DATA DIVISION.
000140 WORKING-STORAGE SECTION.
000150 01 INPUT-TEXT PIC X(8).
000600 PROCEDURE DIVISION.
000610 MAIN-PARAGRAPH.
000620     ACCEPT INPUT-TEXT.
000630     DISPLAY 'Hello world!'.
000640     DISPLAY INPUT-TEXT.
000650     STOP RUN.
```

In command prompt:

```
GoodBye.
Hello world!
GoodBye.
```

Your input

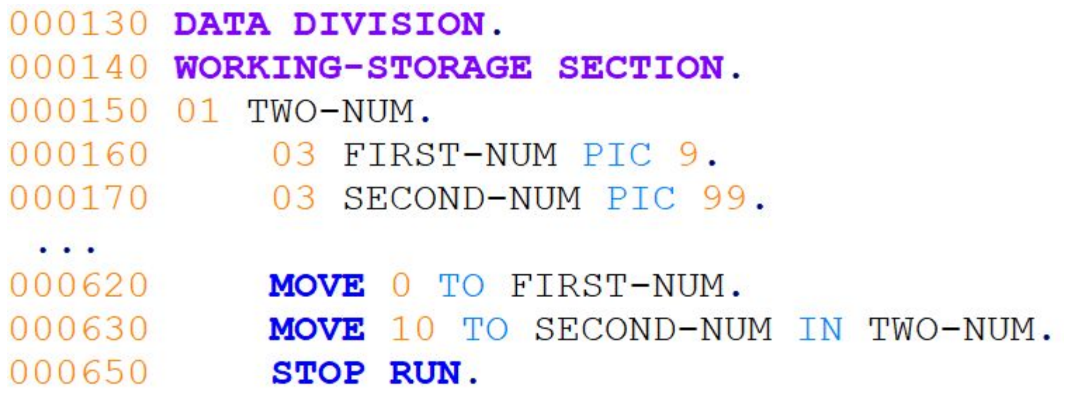
Program output

Assignment Statement

Syntax:

MOVE [value/identifier] TO [identifier]

Example:



```
000130 DATA DIVISION.
000140 WORKING-STORAGE SECTION.
000150 01 TWO-NUM.
000160     03 FIRST-NUM PIC 9.
000170     03 SECOND-NUM PIC 99.
 ...
000620     MOVE 0 TO FIRST-NUM.
000630     MOVE 10 TO SECOND-NUM IN TWO-NUM.
000650     STOP RUN.
```

Arithmetic Statement

Only simple arithmetic

- ADD [value] TO [value] GIVING [variable]
- SUBTRACT [value] FROM [value] GIVING [variable]
- MULTIPLY [value] BY [value] GIVING [variable]
- DIVIDE [value] BY [value] GIVING [variable]

General

- COMPUTE [variable] = [arithmetic expression]
    - Operators: +, -, *, /, **

Selection Statement

Logical operators : NOT, AND, OR

Relational operators : =, >=, <=, >, <, NOT =

Don't add period at the end of all statements here.

IF, Nested IF :

```
IF [condition]
   [statements]
END-IF.
```

```
IF [condition]
    IF [condition]
       …
    END-IF
END-IF.
```

IF-ELSE :

```
IF [condition]
   [statements]
ELSE
   [statements]
END-IF.
```

Switch case statement:

```
EVALUATE [identifier]
   WHEN [value] [statements]
   WHEN [value] [statements]
   ...
END-EVALUATE.
```

The only period for whole IF-statement block.

NOT allowed to use in the assignment

Iteration Statement (NOT allowed to use)

- While loop

```
PERFORM UNTIL [condition]
…
END-PERFORM.
```

- Repeat loop

```
PERFORM WITH TEST AFTER
    UNTIL [condition]
…
END-PERFORM.
```

- For loop

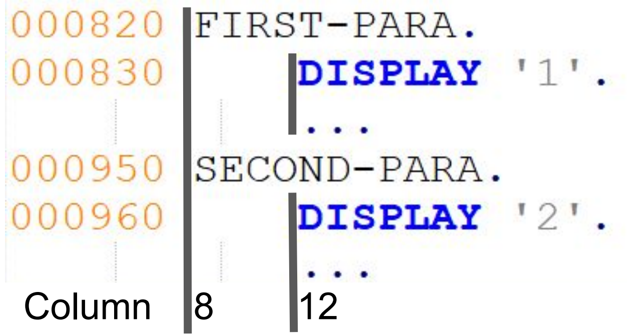
```
PERFORM [value] TIMES
…
END-PERFORM.
```

- For loop

```
PERFORM VARYING [counter variable]
    FROM [initial value] BY [step value] UNTIL [condition]
…
END-PERFORM.
```

Paragraph

- Subroutine in COBOL
- No parameter passing, only global variables
- The scope ends until the occurrence of next paragraph.
- Syntax



```
000820 FIRST-PARA.
000830     DISPLAY '1'.
           ...
000950 SECOND-PARA.
000960     DISPLAY '2'.
           ...
```

- GO TO [paragraph-name]   (keep going, not return to caller)
- PERFORM [paragraph-name]   (return to caller)

Next week tutorial

More on COBOL

1. Sequential File
   a. SELECT-ASSIGN
   b. File description(FD)
   c. File status
2. File I/O
   a. OPEN, CLOSE
   b. READ, WRITE
3. Sorting in a file
   a. Sort file description(SD)
   b. INPUT PROCEDURE, RELEASE
4. String

My Suggestion

- Read it before next week tutorial
- Write some COBOL codes and try to read the files in the assignment

Learning Resource

- Tutorialspoint
  - https://www.tutorialspoint.com/cobol/
- Mainframetechhelp
  - http://www.mainframestechhelp.com/tutorials/cobol/
- COBOL course
  - http://www.csis.ul.ie/cobol/

Google

is your best friend.