

Compiler Construction

Programming Assignment 2

Syntactic and Semantic Definitions for μC

NCKU
National Cheng Kung University



What to do in this Assignment?

- Write an LALR(1) parser for μC using Lex and Yacc.
- The parser supports print I/O, arithmetic operations, and some programming language basic concepts.
- The spec of μC is available for your reference.
- You need to design grammar for your own parser by following the given spec.
- You also need to check semantic correctness by implementing *symbol table*.



Assignment Requirements

- The total score is 105pt. (“0”:0 pt, “1”:30pt, “2”:50pt, …)
- You can judge your code locally with the attached judger.

```
================+==================================================
         Sample | Accept
================+==================================================
 in01_arithmetic | ✓
================+==================================================
 in02_assignment | ✓
================+==================================================
in03_declaration | ✓
================+==================================================
 in04_relational | ✓
================+==================================================
    in05_comment | ✓
================+==================================================
    in06_if_else | ✓
================+==================================================
       in07_for | ✓
================+==================================================
      in08_while | ✓                    Your score after judge
================+==================================================
   in09_function | ✓
================+==================================================
      in10_print | ✓
================+==================================================
    in11_overall | ✓
================+==================================================
Correct/Total problems: 11/11
Obtained/Total scores:  105 105
```

```
// "Hard Coding" will get 0pt.
main() {
    result = read(answer_file);
    print(result);
}
```

https://github.com/aben20807/local-judge/tree/master/examples/student



Scoping (cont.)

```
int height = 99;
{
    float width = 3.14;
}
float length;
{
    string length = "hello world";
    {
        int length[3];
    }
    int width = 66;
}
```

Scope Block A

Scope Block B

Scope Block C

Scope Block D



Scoping (cont.)

- A scope block is a set of statements enclosed within left and right braces ({ and }).
- A variable declared in a block is accessible in the block and all the inner blocks of that block, but not accessible outside the block.
- Different inner scope block in same scope block can't see each other.
- You can declare variable with same name in different scope.



Symbol table functions

- **create_symbol**: Create a symbol table when entering a new scope.
- **insert_symbol**: Insert entries for variables declarations.
- **lookup_symbol**: Look up entries in the symbol table.
- **dump_symbol**: Dump all contents in the symbol table of current scope and its entries when exiting a scope.

Note:

- Function names and their parameters can be properly defined by yourself.



Symbol table

```
int height = 99;
{
    float width = 3.14;
}
float length;
{
    string length = "hello world";
    {
        int length[3];
    }
    int width = 66;
}
```

Insert {height} into symbol table (scope level: 0)

Insert {width} into symbol table (scope level: 1)

Insert {length} into symbol table (scope level: 0)

Insert {length} into symbol table (scope level: 1)

Insert {length} into symbol table (scope level: 2)

Insert {width} into symbol table (scope level: 1)



Symbol table (cont.)

```
int height = 99;
{
    float width = 3.14;
}
float length;
{
    string length = "hello world";
    {
        int length[3];
    }
    int width = 66;
}
```

Index: unique in each symbol table
Address: unique in whole program

Dump scope level 1's symbol table:

Index	Name	Type	Address	Lineno	Element type
0	width	float	1	3	-

Dump scope level 0's symbol table:

Index	Name	Type	Address	Lineno	Element type
0	height	int	0	1	-
1	length	float32	1	2	-



Handle semantic error

```
float y;
x += y
y %= 3
```

error:1: undefined: x

error:3: invalid operation: REM_ASSIGN (mismatched types float and int)

Submission

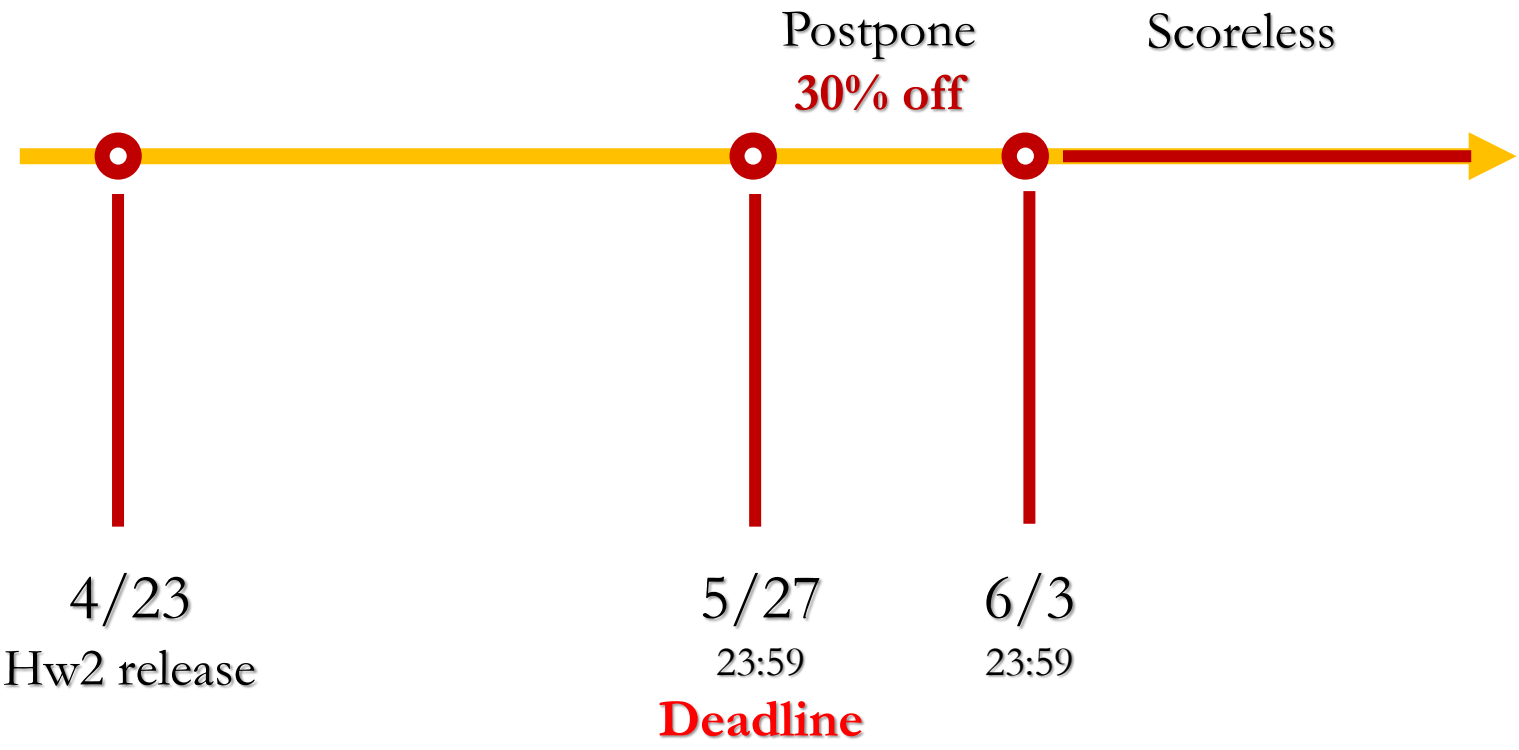
- Upload your homework to Moodle.
- The expected arrangement of your codes:
  - Only **.zip** and **.rar** types of compression are allowed.
  - The directory should be organized as:

```
Compiler_StudentID_HW2.zip/
└── Compiler_StudentID_HW2/
    ├── compiler_hw2.l
    ├── compiler_hw2.y
    ├── common.h
    └── Makefile
```

  - You will lose 10pt if your programs were uploaded in incorrect format!!!



Deadline


Postpone
30% off
Scoreless
4/23
Hw2 release
5/27
23:59
Deadline
6/3
23:59



How to Mail TAs

- Send mail to asrlab@csie.ncku.edu.tw, not any TA's mail!!
- Email subject starts with "[Compiler2021]"



QUESTIONS ?