第九次作业参考答案

1. 考虑文法

$$E \to E+T|T$$
$$T \to T*F|F$$
$$F \to (E)|\mathbf{num}|\mathbf{var}$$

其中**num**表示数字字面量，如123；**var**表示变量，如x。

1. 如果一个项或表达式中不存在变量，它可以在编译期进行求值，例如$1+2*3$。设计一个 SDD 来判断各个项T和表达式E能否在编译期确定它的值。（备注：可以用$E.isconst = \mathbf{true}$表示它可以在编译期求值）
2. 扩展上面的 SDD，使其对表达式进行简单的常量折叠。例如对于$3*2+x+y*(5+3)$，改写为$6+x+y*8$。我们不要求使用交换律等算术规律进行化简，例如不必要化简$3+x+4$

1.

产生式	语义规则
$E \to E_1 + T$	`E.isconst = E1.isconst and T.isconst`
$E \to T$	`E.isconst = T.isconst`
$T \to T_1 * F$	`T.isconst = T1.isconst and F.isconst`
$T \to F$	`T.isconst = F.isconst`
$F \to (E)$	`F.isconst = E.isconst`
$F \to \mathbf{num}$	`F.isconst = true`
$F \to \mathbf{var}$	`F.isconst = false`

2.

$E \to E_1 + T$

```
E.isconst = E1.isconst and T.isconst
if E.isconst:
    E.val = E1.val + T.val
else:
    E.val = E1.val + '+' + T.val
```

$E \to T$

```
E.isconst = T.isconst;
E.val = T.val;
```

$T \to T_1 * F$

```
T.isconst = T1.isconst and F.isconst
if T.isconst:
    T.val = T1.val + F.val
else:
    T.val = T1.val + '*' + F.val
```

$T \to F$

```
T.isconst = F.isconst
T.val = F.val
```

$F \to (E)$

```
F.isconst = E.isconst
if F.isconst:
    F.val = E.val
else:
    F.val = '(' + E.val + ')'
```

$F \to \mathbf{num}$

```
F.isconst = true
F.val = num.val
```

$F \to \mathbf{var}$

```
F.isconst = false
F.val = var.val
```

2. 改写下面的 SDT，消除左递归。其中a、b、c、d为语义动作，不涉及属性计算。0和1是终结符。

$$A \to A\{a\}B|AB\{b\}|0$$
$$B \to B\{c\}A|BA\{d\}|1$$

$$A \to 0A'$$
$$A' \to \{a\}BA'|B\{b\}A'|\varepsilon$$
$$B \to 1B'$$
$$B' \to \{c\}AB'|A\{d\}B'|\varepsilon$$

3. 考虑下图中排版语言的 SDD 和 SDT

	产生式	语义规则
1)	$S \to B$	$B.ps = 10$
2)	$B \to B_1\ B_2$	$B_1.ps = B.ps$ $B_2.ps = B.ps$ $B.ht = \max(B_1.ht, B_2.ht)$ $B.dp = \max(B_1.dp, B_2.dp)$
3)	$B \to B_1\ \mathbf{sub}\ B_2$	$B_1.ps = B.ps$ $B_2.ps = 0.7 \times B.ps$ $B.ht = \max(B_1.ht, B_2.ht - 0.25 \times B.ps)$ $B.dp = \max(B_1.dp, B_2.dp + 0.25 \times B.ps)$
4)	$B \to (\ B_1\ )$	$B_1.ps = B.ps$ $B.ht = B_1.ht$ $B.dp = B_1.dp$
5)	$B \to \mathbf{text}$	$B.ht = getHt(B.ps, \mathbf{text}.lexval)$ $B.dp = getDp(B.ps, \mathbf{text}.lexval)$

	产生式	语义动作
1)	$S \to$ B	$\{ B.ps = 10; \}$
2)	$B \to$ B_1 B_2	$\{ B_1.ps = B.ps; \}$ $\{ B_2.ps = B.ps; \}$ $\{ B.ht = \max(B_1.ht, B_2.ht);$ $B.dp = \max(B_1.dp, B_2.dp); \}$
3)	$B \to$ B_1 **sub** B_2	$\{ B_1.ps = B.ps; \}$ $\{ B_2.ps = 0.7 \times B.ps; \}$ $\{ B.ht = \max(B_1.ht, B_2.ht - 0.25 \times B.ps);$ $B.dp = \max(B_1.dp, B_2.dp + 0.25 \times B.ps); \}$
4)	$B \to$ (B_1)	$\{ B_1.ps = B.ps; \}$ $\{ B.ht = B_1.ht;$ $B.dp = B_1.dp; \}$
5)	$B \to$ **text**	$\{ B.ht = getHt(B.ps, \mathbf{text}.lexval);$ $B.dp = getDp(B.ps, \mathbf{text}.lexval); \}$

1. 修改 SDD，使其包含一个综合属性$B.ch$，表示 Box 中字符的个数。用$getCh(\mathbf{text}.lexval)$表示**text**中字符的个数
2. 将新规则加入 SDT 的合适位置

仅列出新增的部分

1.

产生式	语义规则
$B \to B_1B_2$	$B.ch = B_1.ch + B_2.ch$
$B \to B_1\mathbf{sub}B_2$	$B.ch = B_1.ch + B_2.ch$
$B \to (B_1)$	$B.ch = B_1.ch$
$B \to \mathbf{text}$	$B.ch = getCh(\mathbf{text}.lexval)$

2.

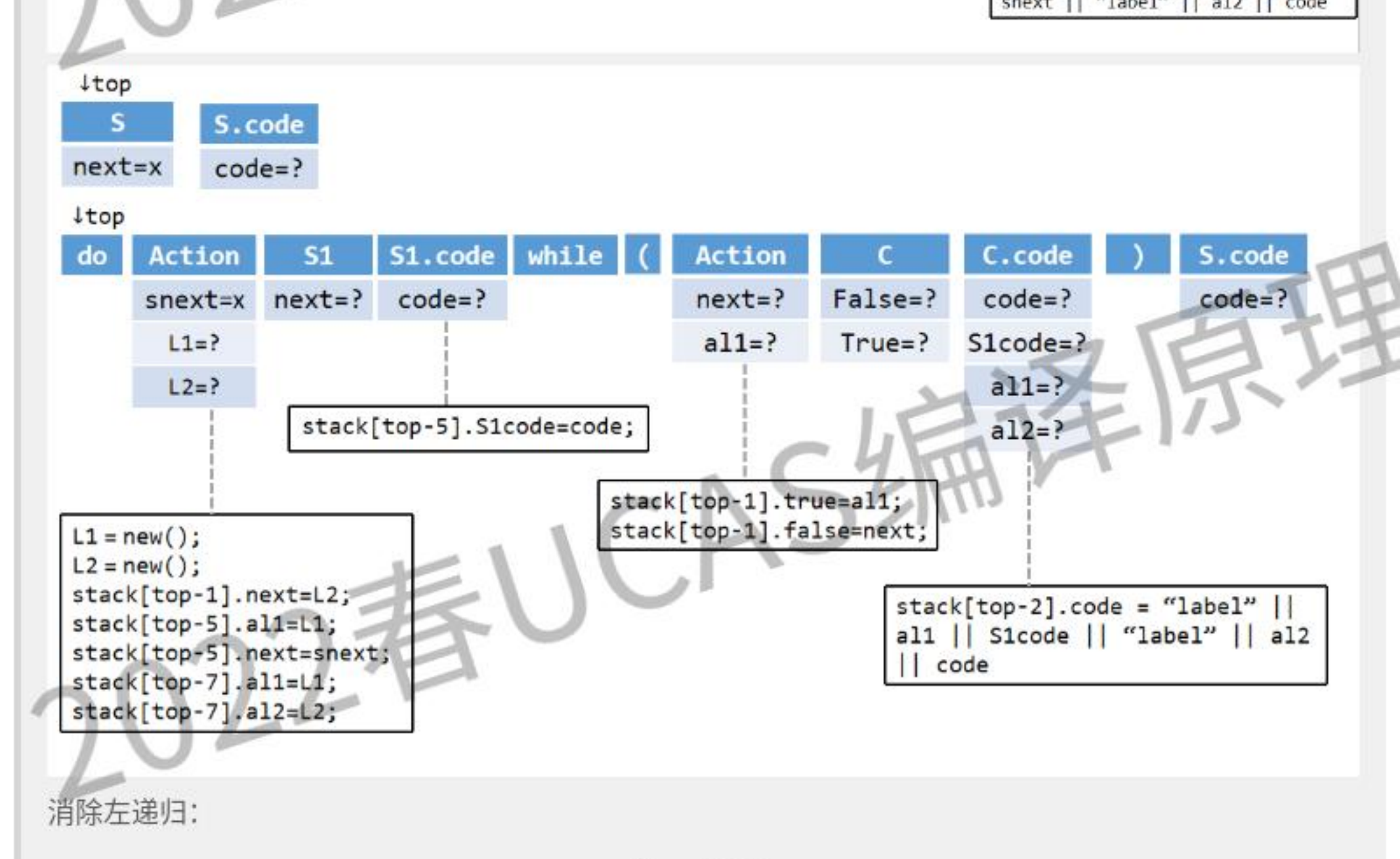
$$S \to B$$
$$B \to B_1B_2 \quad \{B.ch = B_1.ch + B_2.ch\}$$
$$B \to B_1\mathbf{sub}B_2 \quad \{B.ch = B_1.ch + B_2.ch\}$$
$$B \to (B_1) \quad \{B.ch = B_1.ch\}$$
$$B \to \mathbf{text} \quad \{B.ch = getCh(\mathbf{text}.lexval)\}$$

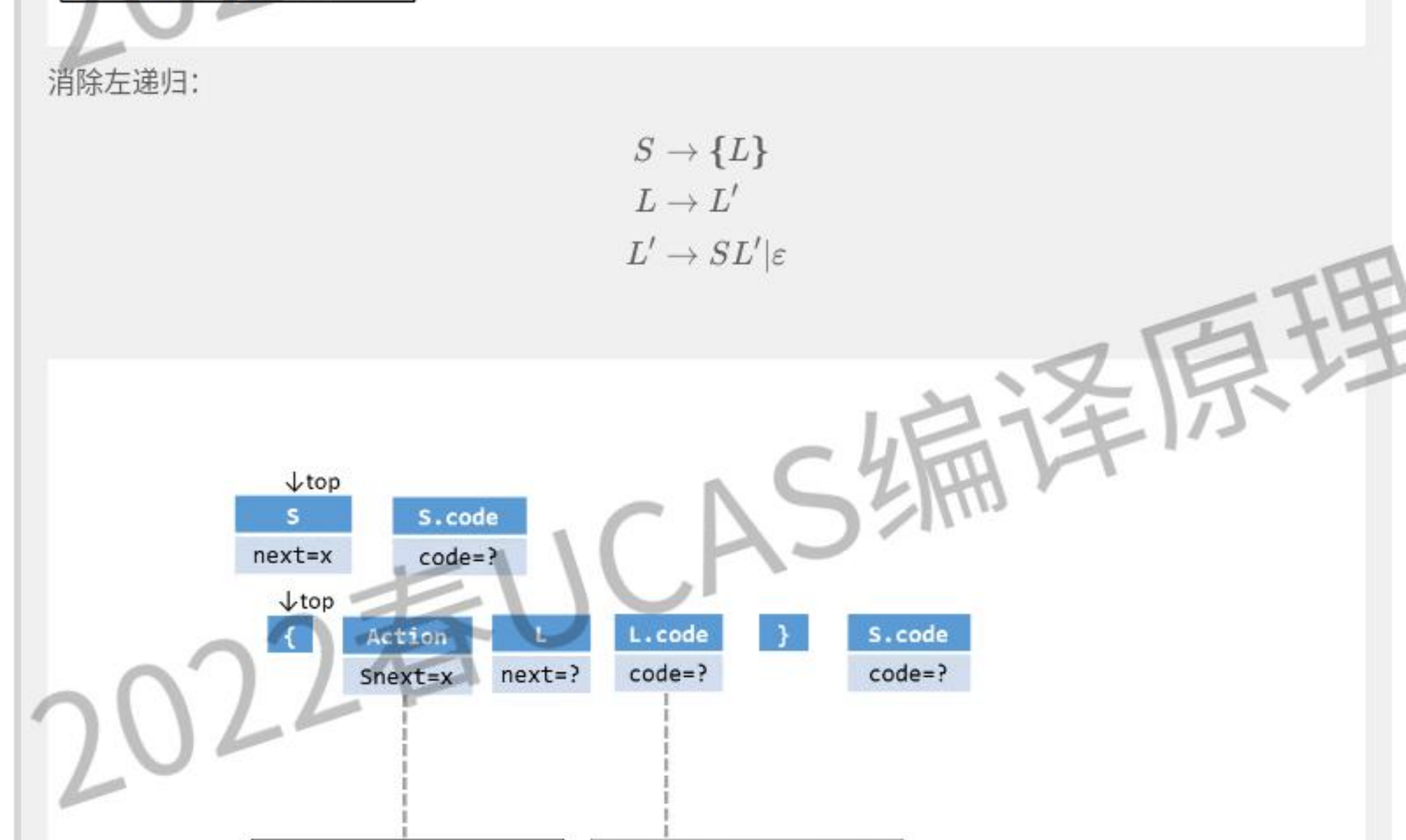
4. 为下面的产生式写出一个和例5.19类似的 L 属性 SDT。这里的每个产生式表示一个常见的 C 语言中那样的控制流结构。你可能需要生成一个三地址语句来跳转到某个标号 `L`，此时你可以生成语句 `goto L`
   1. $S \to \mathbf{if}\ (C) S_1\ \mathbf{else}\ S_2$
   2. $S \to \mathbf{do}\ S_1\ \mathbf{while}\ (C)$
   3. $S \to \{L\}; L \to LS|\varepsilon$

注1：列表中的任何语句都可能包含一条从它的内部跳转到下一条语句的跳转指令，因此简单地为各个语句按顺序生成代码是不够的。
注2：可以先写出SDD，然后按照5.4.5节方法转换为SDT。

	产生式	翻译动作
（1）	S -> **if** (C) S_1 **else** S_2	{L1 = new(); L2 = new(); C.true = L1; C.false = L2;} {S_1.next = S.next;} {S_2.next = S.next;} {S.code = C.code || **label** || L1 || S_1.code || **goto** S.next || **label** || L2 || S_2.code;}
（2）	S -> **do** S_1 **while** (C)	{L1 = new(); L2 = new(); S_1.next = L2} {C.true = L1; C.false = S.next} {S.code = **label** || L1 || S_1.code || **label** || L2 || C.code;}
（3）	S -> '{' L '}'; L -> L_1 S L -> ε	{L.next = S.next;} {S.code = L.code;} {M = new(); L_1.next = M;} {S.next = L.next;} {L.code = L_1.code || **label** || M || S.code;} {L.code = "";}

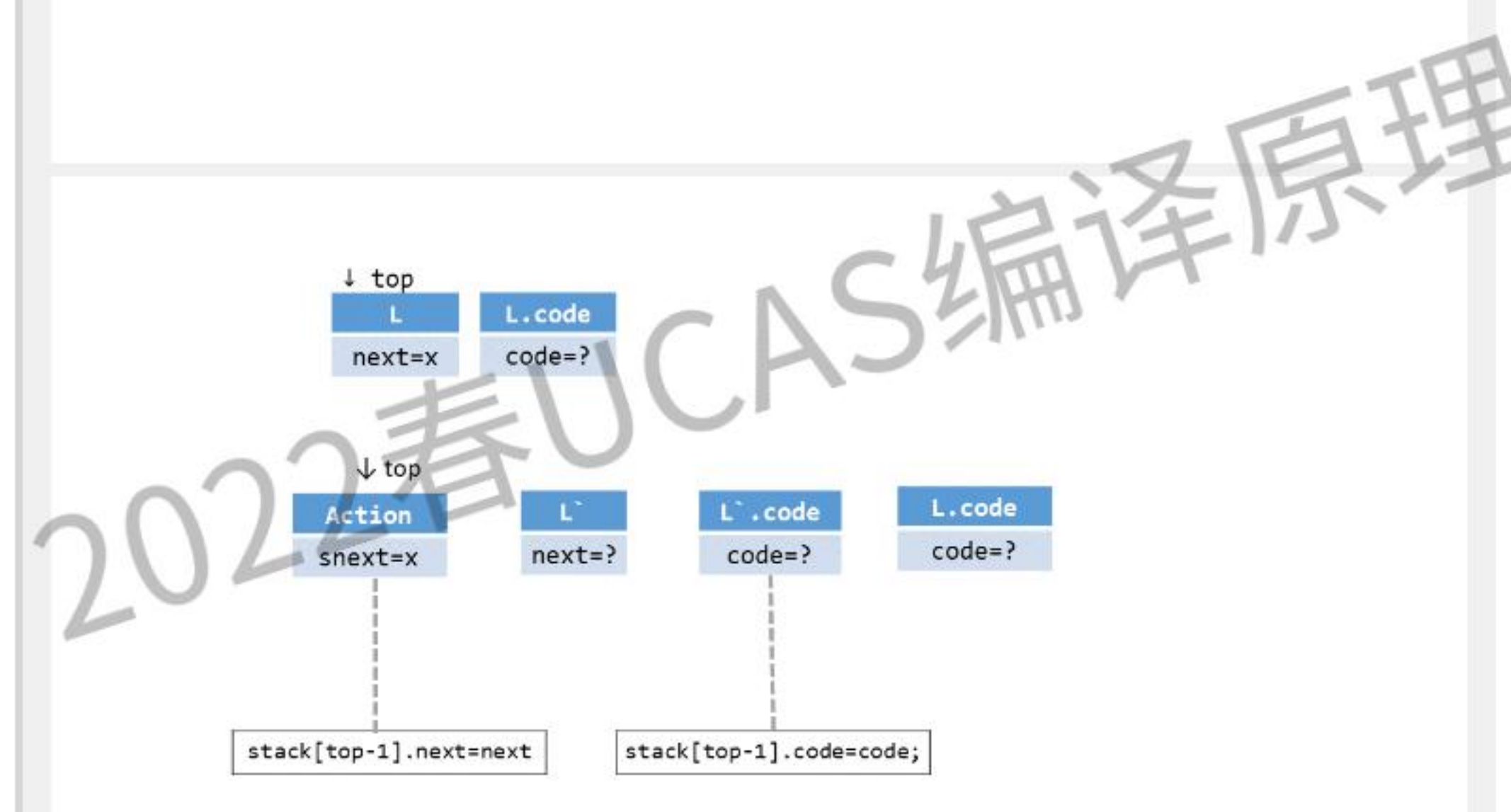
2. 按照5.5.3节的风格，将第1题中得到的每个 SDT 和一个 LL 语法分析器一起实现，但是代码（或指向代码的指针）存放在栈中

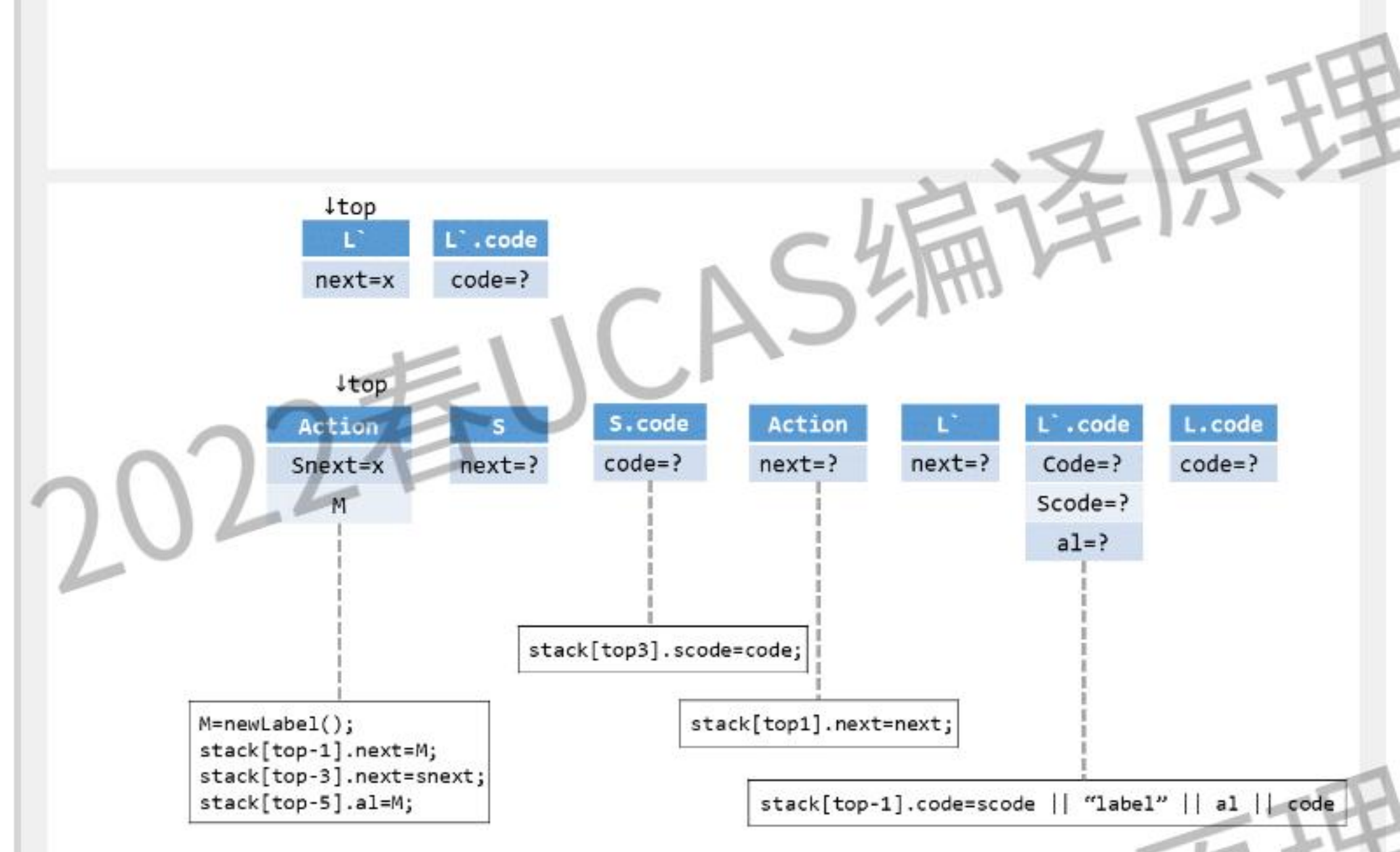


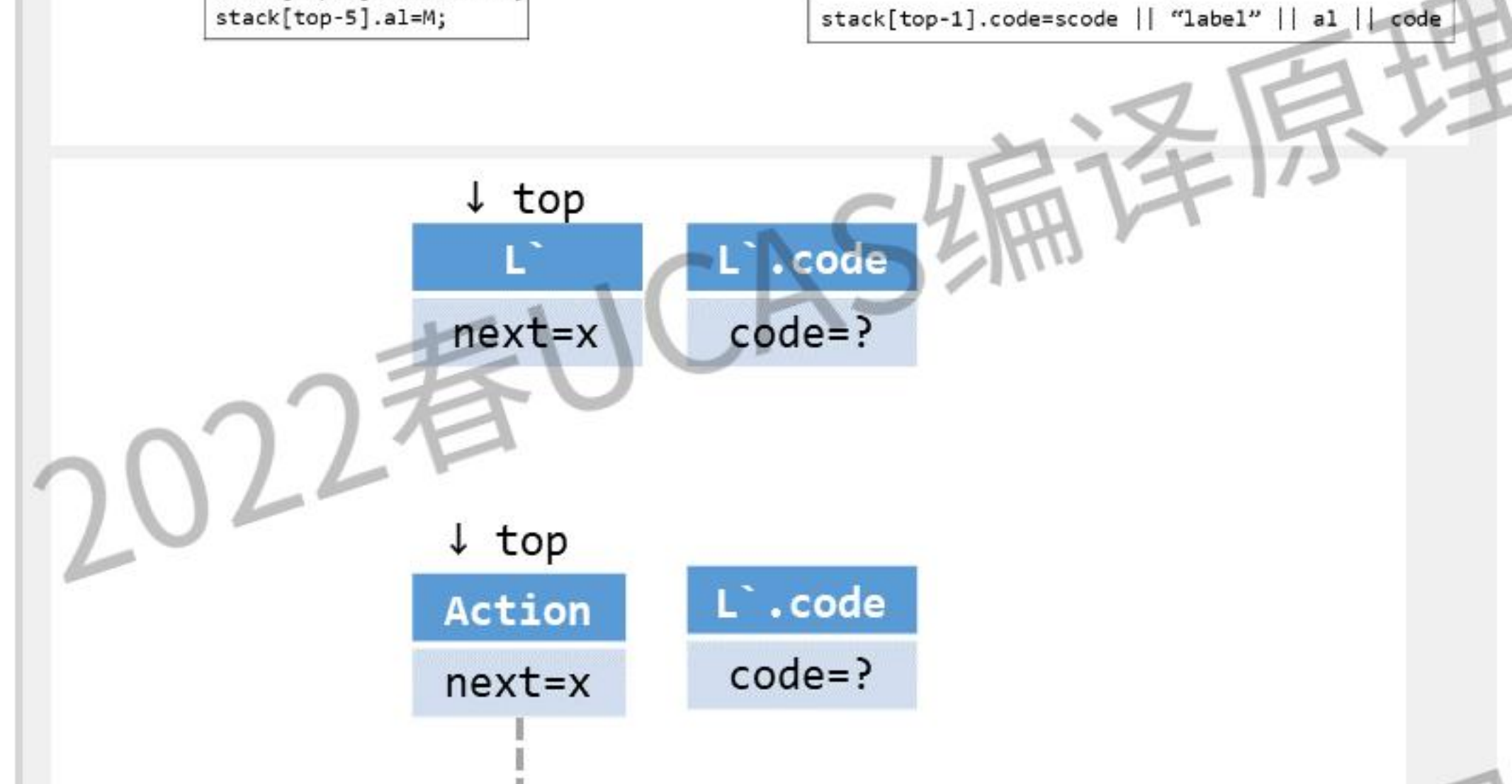


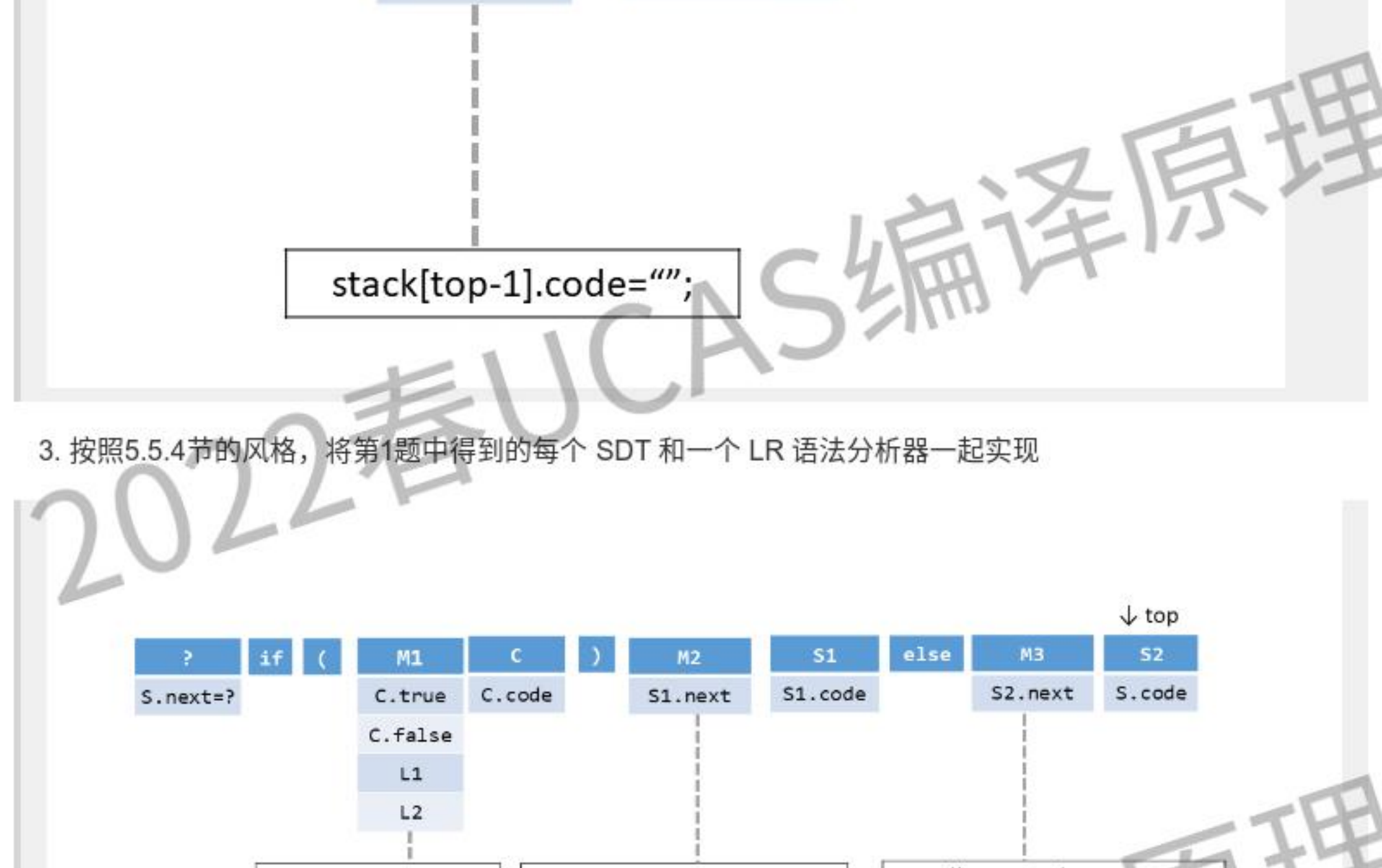
消除左递归：

$$S \to \{L\}$$
$$L \to L'$$
$$L' \to SL'|\varepsilon$$









3. 按照5.5.4节的风格，将第1题中得到的每个 SDT 和一个 LR 语法分析器一起实现