编译原理

第一题，简答题，共 15 分

请简述编译器的工作流程，它每个阶段的输入和输出分别是什么？符号表贯穿编译的各个阶段，请举例说明符号表在编译器每个阶段的作用。

参考答案：

1） 词法分析，语法分析，语义分析和中间代码生成，代码优化，目标代码生成

2） 符号表用来存放语言程序中出现的有关标识符的属性信息，这些信息集中反映了标识符的语义特征属性。在词法分析及语法分析过程中不断积累和更新表中的信息，并在词法分析到代码生成的各阶段，按各自的需要从表中获取不同的属性信息。

①. 收集符号属性。编译程序扫描说明部分收集有关标识符的属性，并在符号表中建立符号的相应属性信息。

②. 上下文语义的合法性检查的依据。同一个标识符可能在程序的不同地方出现，而有关该符号的属性是在这些不同情况下收集的，通过符号表中属性记录可进行相应上下文的语义检查。

③. 作为目标代码生成阶段地址分配的依据。每个符号变量在目标代码生成时需要确定其在存储分配的位置（主要是相对位置），有关区域的标志及相对位置都是作为该变量的语义信息被收集在该变量的符号表属性中。

第二题，简答题，共 15 分

自顶向下分析和自底向上分析都可以处理带有左递归的文法吗？为什么？请分别阐述自顶向下和自底向上的分析方法在分析存在二义性的文法时，可以采取哪些方法处理二义性文法带来的问题？

参考答案：

1） 自顶向下的分析，譬如 LL(1)分析，不能处理左递归文法，要避免文法的左递归，左递归文法会使自顶向下的分析过程陷入无限循环。自底向上的分析，譬如 LR 分析，并非是预测翻译机制，而是使用规约的方式，左递归不会造成问题。

2） 自顶向下的方法，可以采用消除文法中二义性的方法处理二义性带来的冲突，但是并不是所有文法都可以成功消除二义性。自底向上分析方法，以 LR 分析为例，同一个活前缀的两个有效项目可能对应了不同动作，引起了冲突。此时可以通过向前看几个符号解决，也可以通过采用一些语义信息来处理冲突，这些方法都只可以在一定程度上解决二义性引起的冲突。

第三题，简答题，共 15 分

从栈式存储分配的角度对以下程序进行分析：

1）给出以下程序的活动记录（至少给出实参、返回值、控制链、局部数据）；

2）结合活动记录阐述程序中 k 变量的计算过程；

3）给出程序执行后的输出结果。

```
int k = 4;
int q(int n) {
  int k = n + 1;
  return k;
}
int p(int m) {
  if (m > 2) {
    k = k + q(m + 1) + p(m - 1);
  }
  return k;
}
int main() {
  int x = 3;
  p(x);
  printf("k=%d \n", k);
}
```

参考答案：

int k (全局数据)
main
x (局部数据)
int m (参数)
p(3)
~~int n(参数)~~
~~q(4)~~
~~int k (局部数据)~~
int m(参数)
p(2)

在主程序中，首先调用 p(3)，3 作为参数传入，k(1)=k(2)+q(m+1)+p(m-1)，此时全局数据 k 读入的值是 4，随后调用 q(4)，4 作为参数传入，局部数据 k=4+1，返回值为 5，q(4)的记录弹出站；调用 p(2)，2 作为参数，参数＜2，因此返回全局数据 k，返回值为 4，p(2)的记录弹出栈。最后 k=4+5+4，k=13。

第四题，简答题，共 15 分

根据第八章所学知识对以下程序（静态单赋值形式）进行优化，给出每一步优化的结果和所使用的优化方法：

```
j = 1 + 2
t1 = 4 * i
x = a[t1] + 2
t2 = 4 * i
t4 = 5 + a[t2]
t5 = 4 * j
t6 = t5
t7 = t6 * j
```

参考答案：

使用常数合并、公共子表达式消除、常数传播、复写传播等方法进行优化

j = 1 + 2 //常数合并
t1 = 4 * i
x = a[t1] + 2
t2 = 4 * i //公共子表达式消除
t4 = 5 + a[t2]
t5 = 4 * j //常数传播
t6 = t5 //复写传播
t7 = t6 * j
	参考答案：
	t1 = 4 * i
	x = a[t1] + 2
	t4 = 5 + a[t1]
	t7 = 36

第五题，问答题，共 20 分

关于预测分析，考察下述文法：

D → T L

T → **int** | **float**

L → L, **id** | **id**

其中，D 是文法的起始符号，D、T、L 是非终结符，'**int**'、'**float**'、'**id**'、','是终结符。

1）对上述文法消除左递归；

2）针对改造后的文法，请给出每个非终结符号的 FIRST 集合和 FOLLOW 集合；

3）针对改造后的文法，构造预测分析表，并说明该文法是 LL(1)文法吗？为什么？

参考答案：

1） 消除左递归以后，文法如下：

D -> T L

T -> int | float

L -> id L'

L' -> , id L' | ε

2）

FIRST(D)={int, float} FIRST(T)={int, float} FIRST(L)={id} FIRST(L') ={ ',' , ε }

FOLLOW(L') = FOLLOW(L) = FOLLOW(D) = { $ },

FOLLOW(T)={ id }

3）

非终结符	终结符				
	int	float	id	,	$
D	D -> T L	D -> T L			
T	T -> int	T -> float			
L			L -> id L'		
L'				L' -> , id L'	L' -> ε

该文法是 LL(1)文法，因为预测分析表无冲突表项。

第六题，问答题，共 20 分

JSON 是一种轻量级的数据交换语言，在数据的表达格式上遵循 key-value 键值对的形式。其文法的定义如下：

S → { L }

L → L , L | C

C → K : V

K → id

V → id | S

其中，S 是文法的起始符号，S，L，C，K，V 是非终结符，'{'，'}'，','，':'，'id'是终结符。其中，终结符'id'具有属性 lexval 表示其词法值。

请回答以下问题：

1) 该文法能否接受语句{k1:v1, {k2:v2}, k3:v3}？如果能，请给出推导过程；如果不能，请说明理由；

2) 该文法是否具有二义性？为什么？

3) XML 作为一种完整的标记语言，也是一种常见的数据交换语言。XML 一般通过标签嵌套的形式表示待存储的数据。示例如下：某名字为 Mike 的学生编译原理课程成绩为 100，则该信息用 JSON 格式表示为：

```
{  name: Mike,
course: {course_name: compiler, grade: 100}
}
```

对应的 XML 格式为：

```
<name> Mike </name>
<course><course_name> compiler </course_name><grade>100</grade></course>
```

请给出一个语法制导定义，对该文法接受的句子进行翻译，将 JSON 格式转换成 XML 格式。

参考答案：

1）不能，{k2:v2}只可能由产生式 V->S 推导过去，而该语句明显缺少 key 部分，所以不能。

2）该文法具有二义性。考虑语句{k1:v1,k2:v2, k3:v3}，该语句能够被该文法接受，但是存在两棵不同的语法树，所以该文法具有二义性。

3）为每个非终结符引入 code 属性，表示对应的 XML 代码，则 SDD 如下：

产生式	语义动作
S -> { L }	S.code = L.code
L -> L1 , L2	L.code = L1.code || L2.code
L -> C	L.code = C.code
C -> K : V	C.code = '<' || K.code || '>' || V.code || '</' || K.code || '>'
K -> id	K.code = id.lexval
V -> id	V.code = id.lexval
V -> S	V.code = S.code