（若发现问题，请及时告知 ）

1　参考 2.3.4 节采用短路代码进行布尔表达式翻译的 *L*-翻译模式片断及所用到的语义函数。若在基础文法中增加产生式 $E \rightarrow E \uparrow E$，试给出相应该产生式的语义动作集合。其中，“↑”代表“与非”逻辑算符，其语义可用其它逻辑运算定义为 $P \uparrow Q \equiv \text{not}(P \text{ and } Q)$。

参考解答：

$E \rightarrow$ { E_1.false := E.true ; E_1.true := *newlabel* ; } $E_1 \uparrow$
{ E_2.false := E.true ; E_2.true := E.false ; } E_2
{ *E.code := E*$_1$*.code* || *gen* (*E*$_1$*.true* ‘:’) || *E*$_2$*.code* }

2　参考 2.3.5 节进行控制语句翻译的 *L*-翻译模式片断及所用到的语义函数。若在基础文法中增加产生式 $S \rightarrow$ repeat S until E，试给出相应该产生式的语义动作集合。

注：控制语句 repeat <循环体> until <布尔表达式> 的语义为：至少执行 <循环体>一次，直到 <布尔表达式> 成真时结束循环。

参考解答：

$S \rightarrow$ repeat { S_1.next := *newlabel* ; } S_1 until
{ *E*. true := *S*.next ; *E*. false := *newlabel* ; } *E*
{ *S*.code := gen(*E*. false ‘:’) || S_1.code || gen(S_1.next ‘:’) || E.code
}

3　参考 2.3.6 节采用拉链与代码回填技术进行布尔表达式和控制语句翻译的 *S*-翻译模式片断及所用到的语义函数，重复题 1 和题 2 的工作。

参考解答：

$E \rightarrow E_1 \uparrow M\ E_2$　{ *backpatch(E*$_1$*.truelist,M.gotostm) ;*
E.truelist := merge(E$_1$*.falselist, E*$_2$*.falselist) ;*
E.falselist := E$_2$*.truelist*
}

$S \rightarrow$ repeat $M_1\ S_1$ until $M_2\ E$
{ *backpatch(S*$_1$*.nextlist, M*$_2$*.gotostm) ;*
backpatch(E.falselist, M$_1$*.gotostm) ;*
S.nextlist := E.truelist;
}

6.　参考 2.3.6 节采用拉链与代码回填技术进行布尔表达式翻译的 *S*-翻译模式片断及所用到的语义函数。若在基础文法中增加产生式

$E \to \Delta\ (\ E, E\ ,\ E\ )$

其中“Δ”代表一个三元逻辑运算符。逻辑运算 $\Delta\ (\ E_1\ ,\ E_2\ ,\ E_3)$ 的语义可由下表定义：

E_1	E_2	E_3	$\Delta\ (\ E_1\ ,\ E_2\ ,\ E_3\ )$
false	false	false	false
false	false	true	false
false	true	false	true
false	true	true	false
true	false	false	false
true	false	true	false
true	true	false	false
true	true	true	false

试给出相应该产生式的语义处理部分（必要时增加文法符号，类似上述属性文法中的符号 M 和 N），不改变 S-属性文法（翻译模式）的特征。

参考解答：

$E \to \Delta\ (\ E_1\ ,\ M_1\ E_2\ ,\ M_2\ E_3)$
{ *backpatch*(E_1 . *falselist*, M_1 .*gotostm*) ;
backpatch(E_2 . *truelist*, M_2 .*gotostm*) ;
E.truelist := E_3 . *falselist* ;
E.falselist :=*merge*(*merge*(E_1.*truelist*, E_2.*falselist*), E_3 . *truelist* }

9. 参考 2.3.5 节和 2.3.6 节所中关于控制语句（不含 break）翻译的两类翻译模式中的任何一种及所用到的语义函数，给出下列控制语句的一个翻译模式片断：

（*a*）在基础文法中增加关于条件卫士语句的下列产生式

$S \to if\ G\ fi$
$G \to E : S \square G$
$G \to E : S$

注：条件卫士语句的一般形式如

$if\ E_1 : S_1 \square E_2 : S_2 \square \ldots \square E_n : S_n\ fi$

我们将其语义解释为：

(1) 依次判断布尔表达式 E_1 ， E_2 ， …， E_n 的计算结果。

(2) 若计算结果为 true 的第一个表达式为 E_k（$1 \le k \le n$），则执行语句 S_k ；转（4）。

(3) 若 E_1 ， E_2 ， …， E_n 的计算结果均为 false ， 则跳出循环。

(4) 跳出该语句。

（*b*）在基础文法中增加关于循环卫士语句的下列产生式

$$S \rightarrow do\ G\ od$$
$$G \rightarrow E : S \square G$$
$$G \rightarrow E : S$$

注：循环卫士语句的一般形式如

$$do\ E_1 : S_1 \square E_2 : S_2 \square \ldots \square E_n : S_n\ od$$

我们将其语义解释为：

(1) 依次判断布尔表达式 E_1 ， E_2 ， …， E_n 的计算结果。

(2) 若计算结果为 true 的第一个表达式为 E_k（$1 \le k \le n$），则执行语句 S_k ； 转（1）。

(3) 若 E_1 ， E_2 ， …， E_n 的计算结果均为 false ， 则跳出循环。

参考解答：

（*a*）

$S \rightarrow if\ G\ fi$

{

S.nextlist := G . nextlist;

}

$G \rightarrow E\ M_1\ S\ N \square M_2\ G_1$

{

backpatch(E.truelist, M_1 .gotostm) ;

backpatch(E.falselist, M_2 .gotostm) ;

G . nextlist := merge(S . nextlist, merge(N . nextlist, G_1 . nextlist))

}

$G \rightarrow E\ M\ S$

{

backpatch(E . truelist, M .gotostm) ;

G . nextlist := merge(S . nextlist, E . falselist)

}

（*b*）

$S \rightarrow do\ M\ G\ od$

{

backpatch(G .beginlist, M . gotostm) ;

$S.nextlist := G\ .\ nextlist;$

}

$G \rightarrow E : M_1\ S\ N\ \square\ M_2\ G_1$

{

$backpatch(E.truelist, M_1\ .gotostm)\ ;$

$backpatch(E.\ falselist, M_2\ .gotostm)\ ;$

$G\ .beginlist := merge(S\ .\ nextlist, merge(N\ .\ nextlist, G_1\ .\ beginlist))$

$G\ .\ nextlist := G_1\ .\ nextlist;$

}

$G \rightarrow E : M\ S\ N$

{

$backpatch(E.truelist, M\ .gotostm)\ ;$

$G\ .beginlist := N\ .\ nextlist\ ;$

$G\ .\ nextlist := E\ .\ falselist\ ;$

}