第二次作业

1. 判断题：（0.6 分）

(1) **F** 语法制导定义中某文法符号的一个属性，既可以是综合属性，又可以是继承属性。

(2) **T** 把 L-属性定义变换成翻译模式，在构造翻译程序的过程中前进了一大步。

(3) **T** 翻译模式既适于自顶向下分析，又适于自底向上分析。

(4) **T** 用于自顶向下分析的翻译模式，其基础文法中不能含有左递归。

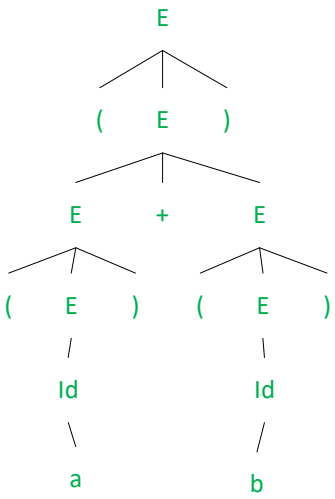
(5) **F** 在基础文法中增加标记非终结符不会导致新的语法分析冲突。

(6) **F** PASCAL 中，由于允许用户动态申请与释放内存空间，所以必须采用栈存储分配技术。

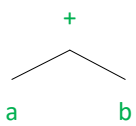
2. 建立表达式((a)+(b))的分析树和语法树（0.4 分）

文法 G：

E → E + E | E * E | (E) | -E | **id**



语法分析树



抽象语法树

3. 按编译中语法制导的生成过程，写出如下流程语句的全部四元式序列：（0.5 分）

if A>B then while A>C do A:=A-B

else if D and F then A:=A+100

三地址码：

100: if A>B goto 102
101: goto 107
102: if A > C goto 104
103: goto 111
104: t1 = A - B
105: A = t1
106: goto 102
107: if D==0 goto 111
108: if F==0 goto 111
109: t2 = A + 100
110: A = t2
111: ...

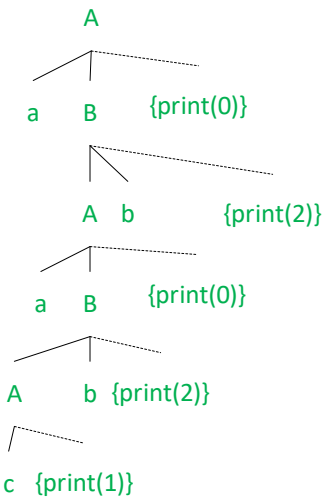
四元式序列：

100: (j>,A,B,102)
101: (j,_,_,107)
102: (j>,A,C,104)
103: (j,_,_,111)
104: (-,A,B,t1)
105: (=,t1,_,A)
106: (j,_,_,102)
107: (jz,D,_,111)
108: (jz,F,_,111)
109: (+,A,100,t2)
110: (=,t2,_,A)
111: ...

4. 在一个移进-归约的分析中采用以下的语法制导翻译方案，在按照某一产生式归约时，立即执行括号中的动作。当分析器的输入为 aacbb 时，打印的字符串是什么？并画出其语法树。（1 分）

A→aB {print “0”;}
A→c {print “1”;}
B→Ab {print “2”}

带有语义动作的语法树如下：



打印的字符串为：12020

5. 为下面的类型写出类型表达式：（0.6 分）

(1) 指向实数的指针数组，数组的下标为 1~100
array (1…100, *pointer* (*real*))
(2) 元素是整型的二维数组（即数组的数组），它的行下标为 0~9，列下标为-10~10

array (0…9, array (-10…10, *integer*)

(3) 一个定义域是从整数到整数指针的函数，它的值域是由一个整数和一个字符组成记录

假定记录中的两个域名为 i 和 c

(*integer* → *pointer* (*integer*)) → *record* ((*i* × *integer*) × (*c* × *char*))

6. 假定有下列 C 语言的声明：（0.4 分）

```
typedef struct {
    int a,b;
} CELL, *PCELL;
CELL foo[100];
PCELL bar(x,y)int x; CELL y; {...}
```

为类型 foo 和 bar 写出类型表达式

foo 的类型表达式是：

array (100, *record* ((*a* × *integer*) × (*b* × *integer*)))

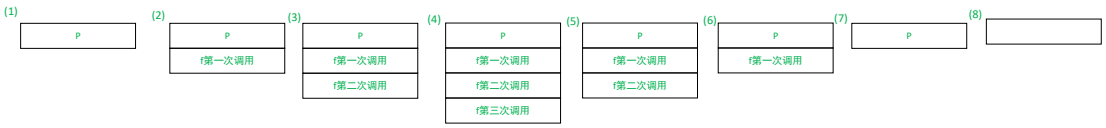
bar 的类型表达式是：

(*integer* × *record* ((*a* × *integer*) × (*b* × *integer*))) → pointer (*record* ((*a* × *integer*) × (*b* × *integer*)))

7. 已经如下 PASCAL 程序，画出它的控制栈中活动记录的变化情况（1 分）

```
program p(input output);
  var a: integer;
  function f(n:integer);
    begin if n=1 then f:=1
                 else f:=n*f(n-1);
    end;
    begin
    a:=3;
    writeln(f(a));
    end
```

控制栈中活动记录简要表示如(1)-(8)所示：



8. 什么叫程序的“基本块”？如何划分一个三地址中间代码程序的“基本块”？说明以下三地址代码程序完成了什么操作？并以它为例，划分其基本块。（1 分）

(1) result:=0

(2) read m
(3) read n
(4) if m<n goto (8)
(5) m:=m-n
(6) result:=result+1
(7) goto (4)
(8) write result
(9) halt

基本块是指程序中一个顺序执行的指令序列，它的第一个指令是这个基本块的入口，最后一个指令是这个基本块的出口，即：

1. 控制流只能从第一个指令进入
2. 除了基本块最后一个指令，控制流不会跳转/停机

划分基本块的方法如下：

首先，确定首指令（基本块的第一个指令），即符合以下任一条的指令：

1. 中间代码的第一个三地址指令
2. 任意一个条件或无条件转移指令的目标指令
3. 紧跟在一个条件/无条件转移指令之后的指令

然后，确定基本块，即：每个首指令对应于一个基本块：从首指令（包含）开始到下一个首指令（不含）。

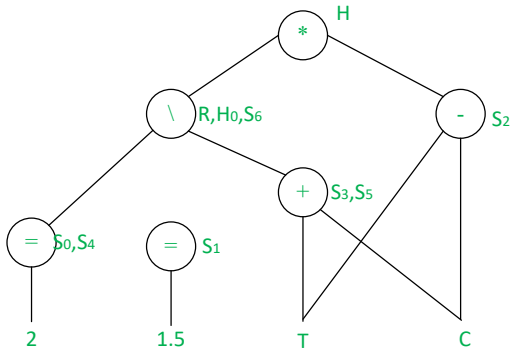
该三地址代码序列完成了根据输入的 m 和 n，以减法的文法计算并输出 m 与 n 的商

该三地址代码序列的基本块划分为： {(1),(2),(3)} {(4)} {(5),(6),(7)} {(8),(9)}

9. 设有如下的程序基本块 P：（1 分）

S0:=2
S1:=3/S0
S2:=T-C
S3:=T+C
R:=S0/S3
H:=R
S4:=S0
S5:=T+C
S6:=S4/S5
H:=S6*S2

(1) 请应用 DAG 图进行优化。



S_0 := 2
S_4 := 2
S_1 := 1.5
S_2 := T-C
S_3 := T+C
S_5 := S_3
R :=2/ S_3
S_6:=R
H := := S_6 * S_2

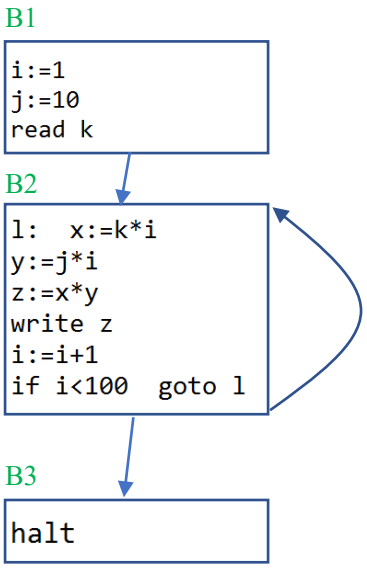
(2) 假定只有 R 和 H 在基本块出口是活跃的，请写出 P 的优化后的四元式序列。

三地址码	四元式序列
100: S_2=T-C	100: (-, T, C, S_2)
101: S_2=T+C	101: (+, T, C, S_3)
102: R = 2/ S_3	102: (/, 2, S_3, R)
103: H = R* S_2	103: (*, R, S_2, H)

10. 试对下面的程序段进行尽可能多的优化，并指明你进行了何种优化，给出优化过程的简要说明及每种优化后的结果形式。（1 分）

```
i:=1
j:=10
read k
l:    x:=k*i
y:=j*i
z:=x*y
write z
i:=i+1
if i<100    goto l
halt
```

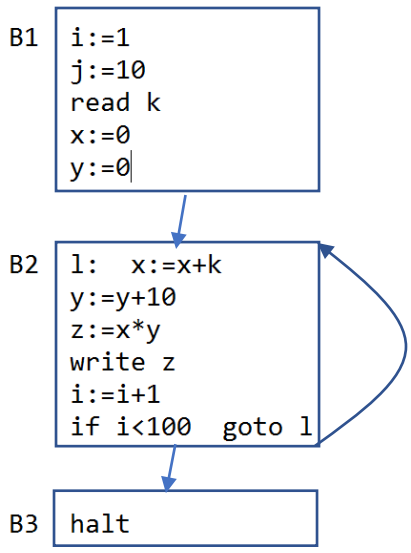
首先，对程序进行基本块划分，如下：



其次，对其进行优化。

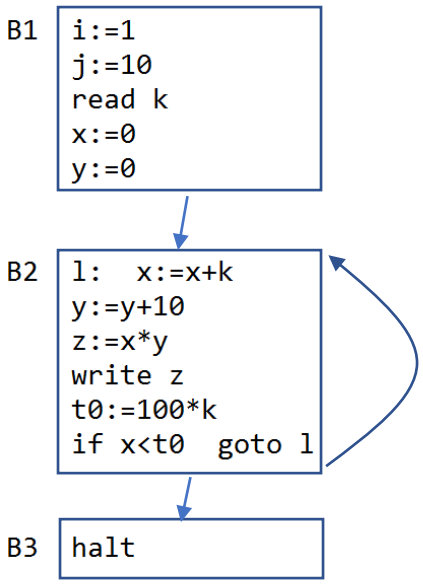
1)强度削弱

由于基本块 B2 是一个循环，由于 i 每次循环增加 1，j 每次循环不变，为常量 10，则在指令 x:=k*i 和指令 y:=j*i 中，x 每次循环增加 k，y 每次循环增加 10，因此，在这两个指令中运算的强度可以由原来的运算乘削弱至运算加，则进行强度削弱后优化的指令序列如下：



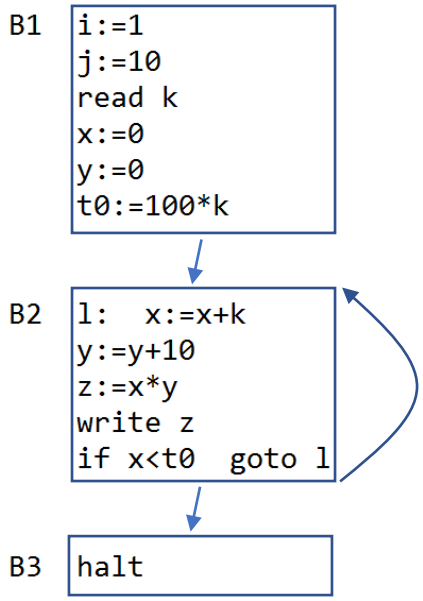
2)删除归纳变量

根据指令 i:=i+1 可知，i 是基本的归纳变量，而 x,y 与 i 保持线性关系，同样也是归纳变量，i<100 可用 x<100*k 或 y<1000 来替换。因此，删除归纳变量 i 后的优化如下：



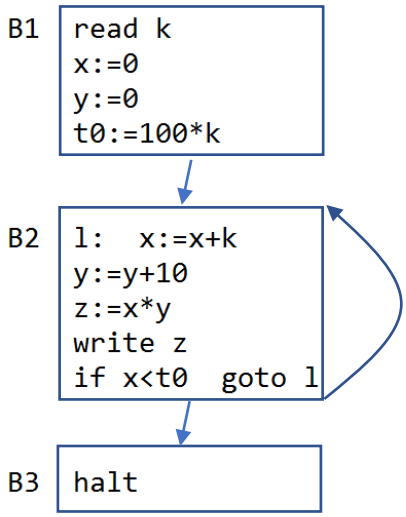
3)代码外提

将循环体中不变的计算或是不变量提到循环体外，放在循环入口处。其中，基本块 B2 中不变运算有 t0:=100*k，优化后为：



4)删除死代码

将流图中因优化而变成的死代码删除，包括 i=1,j=10，优化后为：



附加题

1. C 语言的 for 语句有下列形式，构造一个翻译方案，把 for 语句翻译成三地址代码（1 分）

for (e1;e2;e3) stmt

它和

e1;

while(e2) do begin

    stmt;

    e3;

end

有同样的含义。

S→for(e_1;M_1 e_2;M_2 e_3)N stmt { gen('goto' M_2. instr);
backpatch(stmt.nextlist, M_2.instr);
backpatch(e_2.truelist, N.instr);
backpatch(N.nextlist, M_1.instr);
S.nextlist = e_2.falselist;}

M→ε { M.instr = nextinstr;}

N→ε { N.nextlist = makelist(nextinstr);
gen ('goto_');
N.instr = nextinstr;}