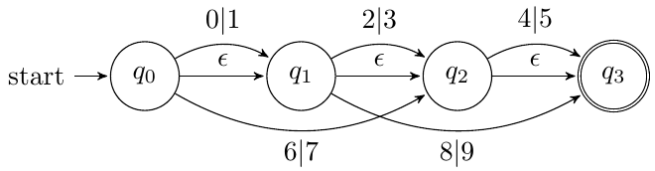
北京大学信息科学技术学院试题

考试科目：编译原理 姓名：_____ 学号：_____

考试时间：2021 年 6 月 30 日 任课教师：梁云

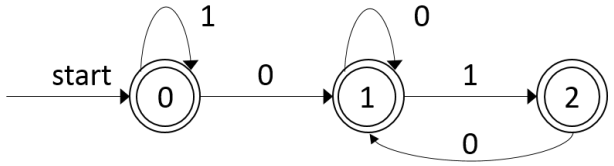
一．单项选择题（每小题 2 分）

1. 下图 NFA 表示的语言中有多少个不同的非空字符串？



A. 37
B. 38
C. 39
D. 40

2. 下面的哪个串可以被图中 DFA 接受



A. 1110001
B. 0110001
C. 1110011
D. 0000110

3. 下面关于间接三元式表示法的描述正确的是

A. 采用间接码表，便于优化处理
B. 不便于表的修改，又增加了存储空间的使用
C. 既便于优化处理，又可以节省存储空间

D. 不便于优化处理，但可以节省存储空间

4. 考虑下面的文法

S → A(S)B | ε

A → S | SB | x | ε

B → SB | y

下面哪个条目不在 SLR 分析状态机的初始集合中？

A. S → •

B. A → •x

C. B → •SB

D. A→ •SB

5. 下面为语言 $L=a^mb^n(m > n >= 0)$设计的文法中，哪一个是 LR(1)的？

A. S → aSb | aS | a

B. S → AB

A → aC

C → aC | ε

B → aBb | ε

C. S → AB

A → Aa | a

B → aBb | ε

D. S → aS | aA

A → aAb | ε

6. 下列说法正确的是

A. LR(k)中的 R 代表从右向左分析。

B. SLR(1)文法都是 LALR(1)文法。

C. 每个文法都可以通过消除左递归和提取左公因子改写为 LL(1)文法。

D. LALR(1)在合并 LR(1)的同心项集时，可能导致移进-规约冲突

7. 下列选项中说法中正确的有：

① 表达能力：NFA > 正则表达式 > DFA

② 任意一个正则表达式可以用一个上下文无关文法表示，且该文法的每个产生式形如“X -> aY”；其中 X, Y 表示任意非终结符（X 和 Y 可能相同），a 表示任意终结符。

③ 正则表达式可以表达语言$\{a^n b^n \mid n = 1,2,3,4\}$。

④ 正则表达式不能表达语言$\{a^m b^n \mid m \neq n\}$。

⑤ 正则表达式 b*(a|ε)(ab|ε)(a|b)*和(b*(a|ε)b*)|(a(a|b)*)表达的是同一种语言。

A. ②④
B. ③④⑤
C. ①⑤
D. ③④

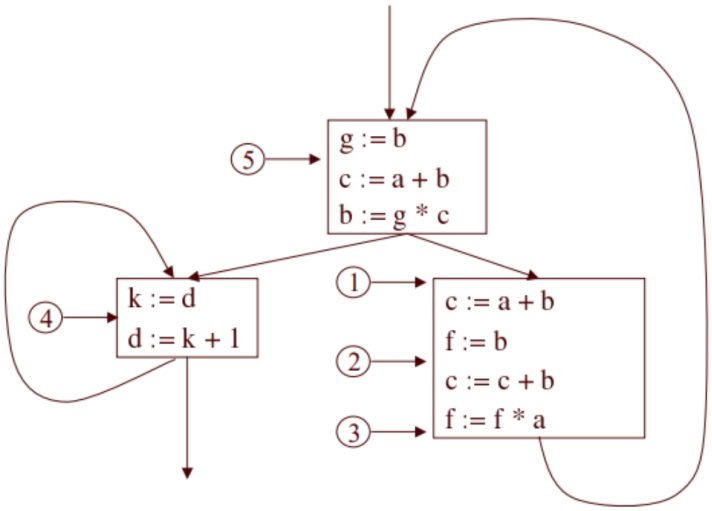
8. 考虑如下的文法：

S → A + B | B - A
A → C = F | E * D | D / C | B (F)
B → C * F | F / C
C → 0 | F & B
D → 1 | C % F
E → 2 | D ^ C
F → 3 | 4 F

在该文法的基础上添加如下哪一个产生式不会引入左递归？

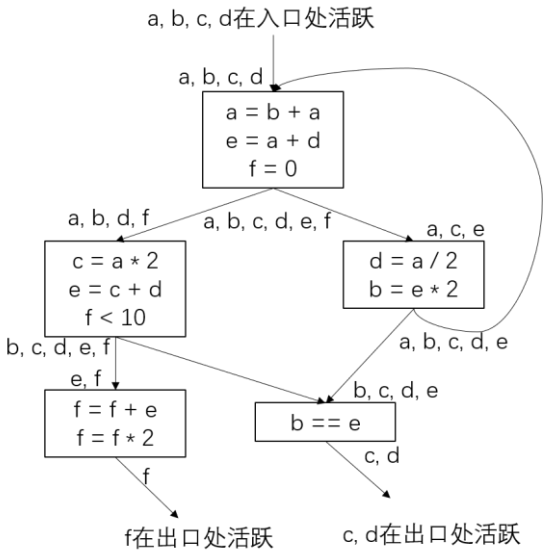
A. F → E ~ F
B. E → S * E
C. D → F [E]
D. B → A (C)

9. 在如下控制流图中，假设程序出口处没有活跃变量，入口处每个变量（a, b, c, d, f, g, k）都被初始化为常量，入口处没有可用的子表达式（注：图中涉及的子表达式为有 b, a+b, g*c, c+b, f*a, d, k+1）。则下列说法中错误的是：



A. 图中循环的个数为 2
B. 不能通过常量传播算法分析得出在④处 c 的值为常量
C. ③处的可用子表达式的个数为 4
D. ①处的活跃变量的集合为{a, b, d}

10. 对下图中的变量进行寄存器分配，图中每个方块代表一个基本块，一共有 a,b,c,d,e,f 六个变量，在程序入口处变量 a,b,c,d 是活跃的，程序两个出口处 f 和 c,d 分别是活跃的，每个基本块入口和出口处活跃的变量也标在图中（基本块上方为入口处，下方为出口处），基于图染色算法，以下说法正确的是？



A. 没有溢出情况下，a 和 c 可以共用同一个寄存器

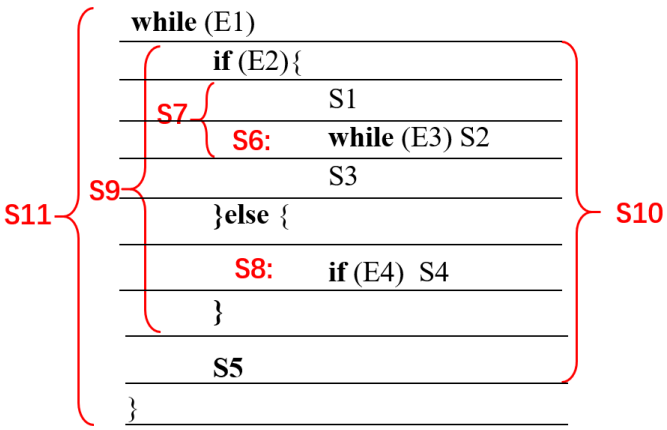
B. f 的 web 有两个

C. 没有溢出情况下，最少可以只使用 3 个寄存器

D. 得到的寄存器冲突图是个完全图

二. 简答题（每小题 6 分）

1. (6 分) 现有如下程序



利用回填技术对程序进行翻译时，给出计算右图中的结构语句的 nextlist 的规则
注意：

- 可以直接使用 E1~E4 的 truelist 和 falselist
- 在求 Sj 的 nextlist 时可以使用 Si 的 nextlist，其中 i < j。

S6.nextlist= _____

S7.nextlist = _____

S8.nextlist = _____

S9.nextlist = _____

S10.nextlist = _____

S11.nextlist = _____

2. (6 分) 对于下面的两个结构体：

```
struct {
    int a;
    char b;
    char c;
} E1;
```

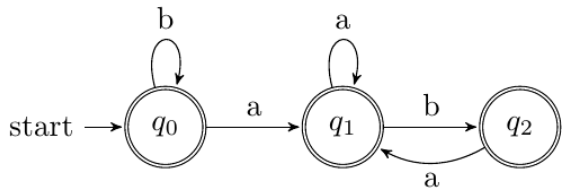
```
struct {
    char a;
    int b;
    char c;
} E2;
```

对于现有变量 int x; E1 y; E2 z; 写出下面这段程序的三地址中间代码：
注意：

- int 为 4 字节，需要按照 4 字节对齐；char 为 1 字节对齐.
- 将结构体的名字视为指向对应结构体起始位置的指针，对该指针进行加法时为字节加法，如 z = y + 1 中 z 指针指的是 y 指向位置后面 1 字节，访问元素时默认可以访问到所需要的字节数，如*y 是默认访问 y.a。

```
S:  while (x < 100){
        if( y.c < z.c){
            x ++;
        } else{
            x = x * 2;
        }
    }
```

3. (6 分) 描述下图 DFA 所接受的语言，并给出一个等价的正则表达式



4. (6 分)寄存器分配：张量是一种多维数组，张量具有形状信息，它们之间的运算包括矩阵乘法（matmul），ReLU 激活，算数运算等。张量运算的程序也可以表达为三地址代码，如下所示，张量 A, B, D, E 在程序开始时是活跃的，出口处 F 是活跃的：

（1） F = 0
（2） C = matmul(A, B)
（3） If avg(C) < 1 goto (5)
（4） goto (9)
（5） A = matmul(C, D)
（6） B = E * 2
（7） F = F + 1
（8） goto (2)
（9） G = ReLU(C)
（10） F = G + F
（11） exit

假设上述代码所有变量都是张量，且维度为 2，各个张量的数据类型都是一致的，我们只关心形状信息，各个张量的形状为

	高度	宽度
A	M	K
B	K	N
C	M	N
D	N	K
E	K	N
F	M	N
G	M	N

其中 M, N, K 都是常量（整型），具体数值未知。

1) 为这段代码划分基本块并标出每个基本块入口和出口处的活跃变量。(3 分)

2) 假设 M=4, N=12, K=3 每个张量都需要被分配物理空间，请结合寄存器分配算法，为每个张量分配一定长度的物理空间（单位为元素数，比如 A 可以分配 M*K=12 个元素大小的空间），假设每个张量都是连续线性存储的，且所有计算步骤都不需要中间临时存储。请给出使用物理存储最小的分配方案以及这个最小的物理空间大小。（3 分，提示，可以先画出冲突图，将没有冲突的张量的物理空间共用）

三. 中间代码生成（12 分）

对于表达式 a = b * -c + d * (b * -c)，画出其对应的语法分析树和 DAG。

四. SDD（10 分）

有如下 SDD：

产生式	语义规则
E → T E’	E’.t = T.t E.e = E’.e
E’→ * T E’$_1$	E’$_1$.t = E’.t + T.t E’.e = E’$_1$.e
E’→ ε	E’.e = E’.t
T → F T’	T’.f = F.f T.t = T’.t
T’ → + F T’$_1$	T’$_1$.f = T’.f * F.f T’.t = T’$_1$.t
T’ → ε	T’.t = T’.f
F → (E)	F.f = E.e
F → digit	F.f = digit.lexval

(1) 在该 SDD 的定义下，若 E →* 1 + (2 * 3 + 4)，则 E.e 的值是多少？（4 分）

(2) 请画出 1 + 2 的注释语法树，并用虚线标注出求值顺序。（6 分）

五．SDT（12 分）

对以下语句：

S→ do S_1 while C

1) 请设计 SDT 实现对该控制结构语句的边扫描边生成（on-the-fly）的翻译过程（4 分）

2) 请设计一个适合于**自底向上**分析的 SDT 以及语法分析器，使用综合属性 code 存储生成的代码, 要求给出对栈顶 top 的地址的相关偏移操作，栈可以用 stack 表示（4 分），并画出使用这个分析器分析题中语法结构时栈内状态（4 分）。在下面的表中，第一行是栈底，部分内容已经给出，答题纸上答案前标明序号①~②。

符号	属性
?	S.next
do	
M	①
S_1	S_1.code
while	
N	②
C	C.code

六．综合题（12 分）

给出一个张量计算文法：

S → L = R

L → T[I]

R → R + F | F

F → F * T[I] | T[I]

I → I, V | V

其中 T 代表张量，V 代表下标变量，都是终结符号。一个根据该文法写出的张量计算例子是

O[i, j] = A[i, k] * B[k, j]

这个例子表达的是矩阵乘法，其中下标变量 k 在右侧出现但是没在左侧出现，表示循环 k 进行的是累加操作。i, j, k 作为下标变量具有循环范围。假设 i, j, k 的范围分别是 M, N, K，那么这个例子翻译为类 C 代码是

```
for (int i = 0; i < M; ++i) {
  for (int j = 0; j < N; ++j) {
    for (int k = 0; k < K; ++k) {
        O[i, j] += A[i, k] * B[k, j]
    }
  }
}
```

1) 请对该文法进行提取左公因子和消除左递归（5 分）

2) 假设我们要求所有的运算和等号只能作用在相同数据类型的操作数上，请对**原文法**设计 SDD 对这一要求进行检查，假设张量的数据类型存放在 type 属性中（3 分）

3) 我们认为张量计算可以分三种类别：Elementwise, Injective, Reduce。其中 Elementwise 是指所用用到的张量都被完全相同的下标索引，例如 O[i, j] = A[i, j] * B[i, j]属于 Elementwise。Reduce 是指含有累加操作的计算，例如上面的矩阵乘法是 Reduce 类型，其余情况认为是 Injective。基于第一问的文法写出适用于自顶向下分析的 SDT 实现对计算类型的分析，分析结果存放在 S 的属性 compute_type 中，请对使用的辅助属性进行说明（4 分）

七．程序优化（10 分）

表达式的复用/冗余消除是编译优化的一个重要环节，而对繁忙表达式的分析就是该环节的一个重要组成部分。

给定一个程序点 p（某个语句执行前或执行后），一个表达式 e，如果在任意一条从 p 开始的路径上，表达式 e 都被求值了，并且在这次求值之前，在相同的路径上 e 没被“杀死”（例如，表达式 x + y 被杀死等价于 x 或 y 被重新赋值），则称 e 为 p 处的一个“繁忙表达式”（very-busy expression）。

(1) 请设计一个数据流算法来找到一个控制流图中每个程序点处的所有繁忙表达式（每个程序点的程序状态设为该点处的繁忙表达式集合）。（4 分，答题纸上标明序号①~④）

只需要说明：

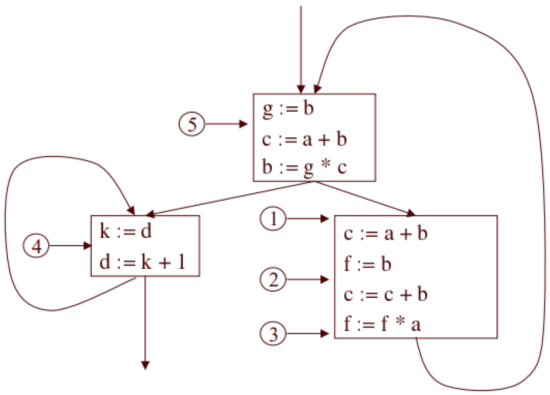
① 数据流算法的方向：正向/反向。

② 程序状态的初始化：在程序入口或出口的程序状态应该初始化为什么值。

③ 状态转移函数：经过一条语句后程序状态应该如何改变。（注：语句 s 中被求值的表达式集合定义为 gen_exp(s)，被杀死的表达式集合定义为 killed_exp(s)）

④ 状态交汇操作：两个基本块的程序状态应该怎样合并。

(2) 给定如下的控制流图，请直接写出程序点②处的繁忙表达式集合。（2 分）



(3) 为了将一个表达式的求值尽可能提前，我们需要计算每个程序点 p 处的“最早放置表达式”。（2 分，答题纸上答案标明序号①~④）

先给出一些概念的简单定义：

活跃语句点：给定程序点 p 和表达式 e，如果某一条从 p 开始的路径上的某一个语句 s 中 e 被求值了，并且在这次求值之前，在相同的路径上 e 没被杀死，则称 s 为 e 在 p 点后的一个“活跃语句点”（live statement point）。e 在 p 点后的所有活跃语句点的集合定义为：LSP(e,p)。

活跃表达式：给定程序点 p 和表达式 e，若 p 点后存在 e 的活跃语句点，则称 e 为 p 点处的一个“活跃表达式”（live expression）。

最早放置表达式：给定程序点 p 和表达式 e，若满足以下两个条件，则称 e 为 p 点处的一个“最早放置表达式”（earliest expression）。

i. 无冗余原则：在 p 处插入语句“v = e”（假设变量 v 与已有变量在命名上无冲突），并且将 p 点后 e 的所有活跃语句点处对表达式 e 的使用都替换为变量 v，而不会引入对 e 的冗余求值（从程序入口到出口所有路径上对 e 的必要的求值次数都不增多）。

ii. 最大覆盖原则：任取一个同样满足条件①程序点 q，若 LSP(e,p) ⊆ LSP(e,q)，则 LSP(e,p) = LSP(e,q)。

下面有若干个程序，请直接写出每个程序中表达式 x + y 是否为该程序的 p 点处的“最早放置表达式”。

```
/* 程序 A */

// p 点
if (...) {
    a=x+y;
} else {
    b=x+y;
}
```

```
/* 程序 B */

z=x+y;
// p 点
if (...) {
    a=x+y;
} else {
    b=x+y;
}
```

```
/* 程序 C */

// p 点
if (...) {
    a=x+y;
} else {
    b=x-y;
}
```

```
/* 程序 D */

x=x+y;
// p 点
if (...) {
    a=x+y;
} else {
    b=x+y;
}
```

程序	x+y 是否为 p 点的最早放置表达式
A	①
B	②
C	③
D	④

(4) 基于第(3)问的定义，为了计算某个程序的程序点 p 处的最早放置表达式集合 earliest[p]，需要如下所列的那些集合？请指出计算所需的集合，并写出 earliest[p] 的计算表达式。（2 分）

i. avail[p]：p 处的可用表达式集合。

ii. busy[p]：p 处的繁忙表达式集合。

iii. live[p]：p 处的活跃表达式集合。