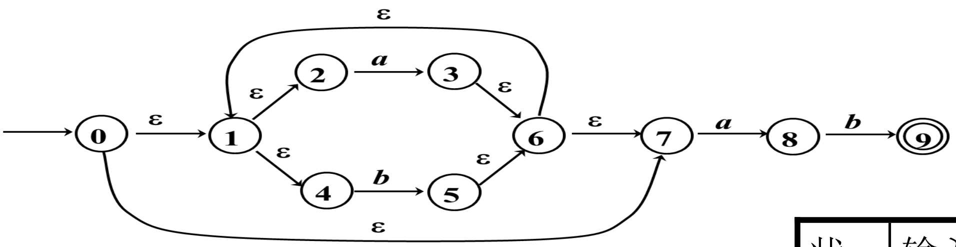
Mid-Term Exam of Compiler Principles (2017）

Name ______________________Student ID______________________

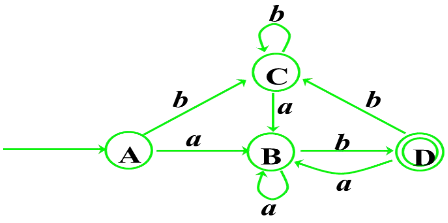
统计：30 人参加考试：<60（2 人） [60,70) (3 人) [70,80)（4 人） [80,90)（13 人） [90,100) （8 人）

注：题目（黑色标注）；答案（红色标注，代码除外）；解析（蓝色标注）

1. Convert the following NFA to DFA (minimum-state DFA, using state minimization algorithm). You need to show the conversion steps. (10 cents)

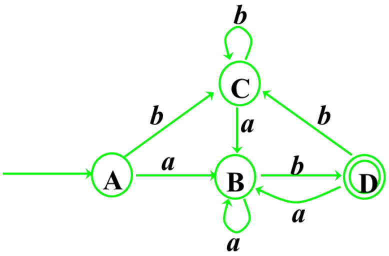


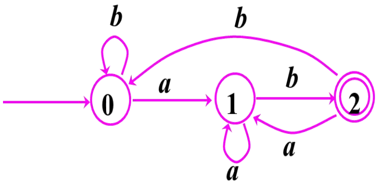
NFA->DFA 思想：将ε闭包合并，得到新的状态（集合）。



状态	输入符号	
	a	*b*
A	*B*	*C*
B	*B*	*D*
C	*B*	*C*
D	*B*	*C*

DFA 最小化思想：把原有状态划分为一些不想交的子集，使得任何两个不同子集的状态是可区别的，而同一子集的任何两个状态是等价的。最后，让每个子集选一个代表，同时消去其他状态。





合并 A,C 状态

解析：

1）部分同学中间状态转化不对，导致后面整体错误；

2）忽略 A,C 状态一样，没有 DFA 最小化；

建议：认真阅读 NFA,DFA 转化章节，通过练习加深理解和掌握。

2. Given the grammar

A → bAaA

A → aAbA

A → ε

a) Give out the leftmost derivation of the string *baba.* (3 cents)

第一种方式：

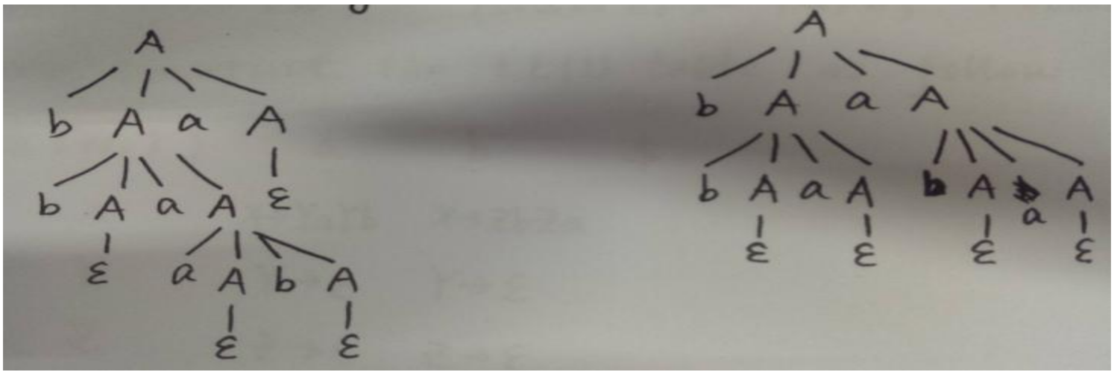
A => bAaA=>baAbAaA=>babAaA=>babaA=>baba

第二种方式：

A => bAaA=>baA=>babAaA=>babaA=>baba

b) Is the grammar ambiguous? Why? (5 cents)

Yes. It has two leftmost derivation and two parse tree for the string baba。



c) What's the language described by the grammar? (2 cents)

It generates the string with equal number of *a* and *b* (including null string).

解析：

1) 本题相对比较简单，但是在第三问有同学会忽略为空的情况；

2) 注意在第二问中，判断是不是 ambiguous 的方式是能够有多个 parse tree 或多个最左（最右）推导；而不是有两个推导就行了。一些同学对于概念理解不彻底，或则考试时候表述不够准确，需要注意。

3. Consider the following grammar

D → T L
T → int | real
L → id R
R → , id R | ε

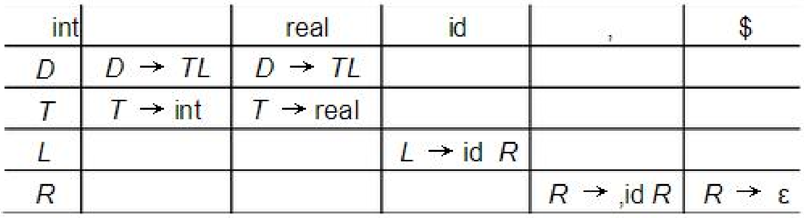
a). Calculate FIRST and FOLLOW set for non-terminals in the grammar, and show the nullable nonterminals. (10 cents)

FIRST(D)={int, real}, FIRST(T)={int, real}, FIRST(L)={id},FIRST(R) ={ , ε}

FOLLOW(R)= FOLLOW(L) = FOLLOW(D) = { $ }, FOLLOW(T)={ id }

R is nullable nonterminal.

b). Construct the LL(1) parsing table for the grammar. Is the grammar a LL(1) grammar? Why? (15 cents)



	int	real	id	,	$
D	*D* → *TL*	*D* → *TL*			
T	*T* → int	*T* → real			
L			*L* → id *R*		
R				*R* → ,id *R*	*R* → ε

It is LL(1) grammar, we can see from the parsing table, it doesn't include any entry that has more than one item.

解析：

1）个别同学不知道 First、Follow 集合怎么求，容易丢掉 FIRST(R)= { , ε}中的空串（建议参考《编译原理与实践》中的 First，Follow 相关部分）；

2）新书中没考虑$的情况，所以同学把 Follow（R/L/D）置为空也算对的；

4. The program 4.1 in the page 89 has presented the recursive-descent interpreter for part of the Grammar 3.15 (see page 53). Please continue to finish the recursive-descent procedure for the remaining part of Grammar 3.15. (10 cents)

```
// 省略掉代码中原有部分，主要针对 S,E
//**********S*****************
int S_Follow[] = {};
int S(void) {
      switch (tok.kind) {
            case ID:   case NUM: case LPAREN: return E();
            default:   print("expected ID, NUM, or left-paren");
                  skipto(S_follow); return 0;
      }
}
//**********E*****************
int E_Follow[] = { RPAREN, EOF };
int E(void) {
            switch (tok.kind) {
            case ID: case NUM: case LPAREN: return Eprime(T());
            default: print("expected ID, NUM, or left-paren");
                  skipto(E_follow);
                  return 0;
      }
}
// Eprime 里面已经包含对于 E'(表示为 E1)的处理
int Eprime(int a) {
      switch (tok.kind) {
            case PLUS: eat(PLUS); return Eprime(a+T());
            case RPAREN: case EOF: return a;
```

```
        default: print("expected PLUS, RPAREN, or EOF");
            skipto(E1_follow);
            return 0;
    }
}
```

解析：

1） 这题相对来说比较简单，但很多同学丢掉了处理 S 的函数；

5. Given the program 5.2 in the page 106.

a) Please describe how to use the hash table to maintain the scope of variables. (5 cents)

Insert when encountering a variable, and delete at the end of the scope of the variable. Specifically, Consider σ+ {a ↦ τ2} when σ contains a ↦ τ1 already. The insert function leaves a ↦ τ1 in the bucket and puts a ↦ τ2 earlier in the list. Then, when pop(a) is done at the end of a's scope, σ is restored.

b) Improve the hash implementation to hide the representation of the *table* type inside an abstract module, so that clients are not tempted to manipulate the data structure directly(only through the *insert*, *lookup*, and *pop* operation) (10 cents)

```
struct abstractTable {
    bucket *table[SIZE];
};
void intersert(string key, void *binding, struct abstractTable aTable) {
    // 将 table 换成 aTable.table
}
void lookup(string key, struct abstractTable aTable) {
    // 将 table 换成 aTable.table
}
void pop(string key, struct abstractTable aTable) {
    // 将 table 换成 aTable.table
}
```

解析：

1）这题属于主观题，第一问重点要突出在变量作用域开始和结束时，Hash 表如何处理。此外需要指出表里已有同名变量时，如何维护变量的作用域。

2）第二问答案不唯一，有的同学用 C++类来处理，也可以。

6. Please answer the following questions.

a) Whether we can use stacks to hold all local variables for higher-order functions or not? Why? (5 cents)

No, they cause local variables to need lifetimes longer than their enclosing function invocations.

b) Describe the function of the following four frame interfaces which are defined in page 136. (10 cents)

(1) F_frame F_newFrame(Temp_label name, U_boolList formals);

- F_frame: contains information of formal parameters and local variables allocated in this frame;
- U_boolList : where *l* is a list of *k* booleans: true for each parameter that escapes and false for each parameter that does not.

(2) Temp_label F_name(F_frame f);

- Name the frame f;

(3) F_accessList F_formals(F_frame f);

- field is a list of *k* "accesses" denoting the locations where the formal parameters will be kept at run time, as seen from inside the callee.
- Parameters may be seen differently by the caller and the callee.

(4) F_access F_allocLocal(F_frame f, bool escape);

- This returns an InFrame access with an offset from the frame pointer.
- The boolean argument to allocLocal specifies whether the new variable escapes and needs to go in the frame; if it is false, then the variable can be allocated in a register.
- The calls to allocLocal need not come immediately after the frame is created. Thus, there will be a distinct temporary or frame slot for every variable declared within the entire function.

解析：

1）这题答案就在书本中，请大家仔细阅读课本。出题本意就是让大家多熟悉课本，注意其中的细节。

7. Implement the following methods defined in page 156 by C language.

a) static T_stm unNx (Tr_exp e) (7 cents)

```
static T_exp unNx(Tr_exp e) {
    switch (e->kind) {
    case Tr_nx:
        return e->u.nx;
    case Tr_cx: {
        Temp_temp r = Temp_newtemp();
        Temp_label t = Temp_newlabel(), f = Temp_newlabel();
        doPatch(e->u.cx.trues, t);
        doPatch(e->u.cx.falses, f);
        return T_Exp(T_Eseq(T_Move(Temp(r), T_Const(1)),
            T_Eseq(e->u.cx.stm,
                T_Eseq(T_Label(f),
                    T_Eseq(T_Move(Temp(r), T_Const(0))
                        T_Eseq(T_Label(t),
                            T_Temp(r)))))));

    }
    case Tr_ex:
        return T_Exp(e->u.ex);
    }
    assert(0);
}
```

b) static struct Cx unCx (Tr_exp e) (8 cents)

```
static T_exp unCx(Tr_exp e) {
    switch (e->kind) {
    case Tr_cx:
        return e->u.cx;
    case Tr_ex: {
        struct Cx condCx;
        condCx.tures = NULL;
        condCx.falses = NULL;
        if (e->u.ex->u.kind == T_CONST) {
            condCx.stm = T_CJump(T_gt, e->u.ex, T_Const(0), NULL, NULL);
            condCx.tures = PatchList(&(condCx.stm)->u.CJUMP.true, NULL);
            condCx.falses = PatchList(&(condCx.stm)->u.CJUMP.false, NULL);
        }
        else {
            condCx.stm = T_Exp(e->u.ex);
        }
        return condCx;
    }
    case Tr_nx:
        assert(0);
    }
    assert(0);
}
```

解析：

1）这题答案不唯一，上面答案仅供大家参考。

2）在构建函数的过程中，需要明确函数输出 T_exp 的功能以及结构，这样才利于转换。

考试总结

1. 本次考试，整体成绩良好。但也存在个别同学由于书本知识掌握不够，导致成绩不理想。希望接下来可以基于书本打牢基础。

2. 本次考试涉及到书中细节的理解，需要平时能够认真看书。否则临时在书中找信息、理解并答题，时间来不及。

3. 在客观题如题 1-4 上，部分同学还存在概念不清晰，理论知识掌握不够的情况。如 NFA 到 DFA 的转换以及最小化技术，ambiguity，First 和 Follow 集合等，需要进一步加强。