1997 年编译原理试题

1．（10 分）某操作系统下合法的文件名为

device:name.extension

其中第一部分（device:）和第三部分（.extension）可缺省，若 device, name 和 extension 都是字母串，长度不限，但至少为 1，画出识别这种文件名的确定有限自动机。

2．（20 分）

a. 下面的二义文法描述命题演算公式，为它写一个等价的非二义文法。

S—> S and S | S or S | not S | p | q | (S)

b. 下面文法是否为 LL(1)文法？说明理由。

S—> A B | P Q x　　A—> x y　　B—> b c

P—> d P | ε　　Q—> a Q | ε

3．（10 分）某些语言允许给出名字表的一个属性表，也允许声明嵌在另一个声明里面，下面文法抽象这个问题。

D —> attrlist namelist | attrlist (D)

namelist —> id, namelist | id

attrlist —> A attrlist | A

A —> decimal | fixed | float | real

D —> attrlist namelist 的含义是：在 namelist 中的任何名字有 attrlist 中给出的所有属性。D—> attrlist (D) 的含义是：在括号中的声明提到的所有名字有 attrlist 中给出的所有属性，而不管声明嵌套多少层。 写一个翻译方案，它将每个名字的属性个数填入符号表。为简单起见，若属性重复出现，则重复计数。

4．（10 分）把表达式

-(a+b)*(c+d)+(a+b+c)

翻译成四元式。

5．（10 分）由于文法二义引起的 LR（1）分析动作冲突，可以依据消除二义的规则而得到 LR（1）分析表，根据此表可以正确识别输入串是否为相应语言的句子。对于非二义非 LR（1）文法引起的 LR（1）分析动作的冲突，是否也可以依据什么规则来消除 LR（1）分析动作的冲突而得到 LR（1）分析表，并且根据此表识别相应语言的句子？若可以，你是否可以给出这样的规则？

6．（5 分） UNIX 下的 C 编译命令 cc 的选择项 g 和 O 的解释如下，其中 dbx 的解释是“dbx is an utility for source-level debugging and execution of programs written in C”。试说明为什么用了选择项 g 后，选择项 O 便被忽略。

-g　　Produce additional symbol table information for dbx(1) and dbxtool(1) and pass -lg option to ld(1) (so as to include the g library, that is: /usr/lib/libg.a). When this option is given, the

-O and -R options are suppressed.

-O[level] Optimize the object code. Ignored when either -g, -go, or -a is used. ...

7．（15 分）下面程序在 SUN 工作站上运行时陷入死循环，试说明原因。如果将第 8 行的 long *p 改成 short *p，并且将第 23 行 long k 改成 short k 后，loop 中的循环体执行一次便停止了。试说明原因。

```
main()
{
    addr();
    loop();

}

long *p;

loop()
{
    long i,j;

    j=0;
    for(i=0;i<10;i++){
        (*p)--;
        j++;
    }
}

addr()
{
    long k;

    k=0;
    p=&k;
}
```

8．（15 分）下面程序的结果是 120。但是如果把第 10 行的 abs(1)改成 1 的话，则程序结果是 1。试分析为什么会有这不同的结果。

```
int fact()
```

```
{
    static int i=5;

    if(i==0) {
        return(1);
    }
    else {
        i=i-1;
        return((i+abs(1))*fact());
    }
}

main()
{
    printf("factor of 5 = %d\n", fact());
}
```

9．（5 分）在文件 stdlib.h 中，关于 qsort 的外部声明如下：

```
extern void qsort(void *, size_t, size_t,
    int (*)(const void *, const void *));
```

下面 C 程序所在的文件名是 type.c，用 Solaris C 编译器编译时，错误信息如下：

```
"type.c", line 24: warning: argument is incompatible with prototype: arg #4
```

请你对该程序略作修改，使得该警告错误能消失，并且不改变程序的结果。

注：程序中关于变量 astHypo 和 n 的赋值以及其它部分被略去。Solaris C 比 ANSI C 有更严格的静态检查。

```
#include <stdlib.h>

typedef struct{
    int         Ave;
    double    Prob;
}HYPO;

HYPO       *astHypo;
int        n;

int HypoCompare(HYPO *stHypo1, HYPO *stHypo2)
{
```

```
        if (stHypo1->Prob>stHypo2->Prob){
                return(-1);
        }else if (stHypo1->Prob<stHypo2->Prob) {
                return(1);
        }else{
                return(0);
        }
}/* end of function HypoCompare */

main()
{
      qsort ( astHypo,n,sizeof(HYPO),HypoCompare);
}
```