# 逻辑题编程实例

翠竹的同学都在某德资企业工作，溪兰是翠竹的大学同学，洞松是该德资企业的部门经理。该德资企业的员工有些来自淮安。该德资企业的员工都曾到德国研修，他们都会说德语。以下哪项可以从以上陈述中得出？（MBA.2013.51）

（A）洞松与溪兰是大学同学。

（B）翠竹的大学同学有些是部门经理。

（C）翠竹与洞松是大学同学。

（D）溪兰会说德语。

（E）洞松来自淮安。

```
work(X,dezi):-
tongxue(cuizu,X).

tongxue(cuizu,xilan).

work(dongsong,dezi).

speak(X,deyu):-work(X,dezi).

answer:-
    tongxue(cuizu,dongsong).


?- answer.
false.
```

```
answer:-
    speak(xilan,deyu).


?- answer.
true.
```

张老师让小李、小王、小赵和小杨四位同学确认徐州会战、南京保卫战、湘西会战、淞沪会战四场战役的先后顺序。

- 小李说: 以上战役依次是第一、第二、第三、第四场战役;
- 小王说: 以上战役依次是第一、第三、第四和第二场战役;
- 小赵说: 以上战役依次是第四、第三、第一和第二场战役;
- 小杨说: 以上战役依次是第四、第二、第三和第一场战役。

张老师说: 小赵一个都没有说对，小李和小王各说对了一个，而小杨说对了两个。 由此可以判断：

- A. 第一场战役是南京保卫战;
- B. 第二场战役是湘西会战;
- C. 第三场战役不是徐州会战;
- D. 第四场战役不是淞沪会战。

# 子集验证函数

```
subset([],_).
subset([X|Y], Z):-
    member(X,Z),
    subset(Y,Z).

?- subset([],[a,b,c]).
true.
```

```
?- subset([c,b],[a,b,c]).
true .

?- subset([c,d],[a,b,c]).
false.

?- subset([a,b,c],[a,b]).
false.
```

# 元素互异验证函数

```
unique([]).
unique([X|Y]):-
    not(member(X, Y)),
    unique(Y).

?- unique([]).
true.
```

```
?- unique([a,a,b]).
false.

?- unique([a,b]).
true.

?- unique([a,b,c]).
true.
```

# 匹配计数函数

```
total([], _, 0).
total([A1|A2], [B1|B2], N):-
    (A1=B1, N1=1; A1\=B1, N1=0),
    total(A2, B2, N2),
    N is N1+N2.

?- total([], _, X).
X = 0.
```

```
?-
total([a,b,d,c],[d,b,a,c],X).
X = 2.

?-
total([a,b,c,d],[d,b,a,c],X).
X = 1.

?- total([a,b,c],[d,b,a,c],X).
X = 1.

?-
total([a,b,d,c],[d,b,a,c],2).
true.
```

# 确定顺序

```
answer:-
    Orders=[_,_,_,_],
    subset(Orders, [1,2,3,4]),
    unique(Orders),
    total(Orders, [1,2,3,4], Li),
    Li=1,
    total(Orders, [1,3,4,2], Wang),
    Wang=1,
    total(Orders, [4,3,1,2], Zhang),
    Zhang=0,
    total(Orders, [4,2,3,1], Yang),
    Yang=2,
    write(Orders).
```

```
?- answer.
[3,2,4,1]
true.
```

# 确定顺序函数

```
getOrders(Orders):-
    Orders=[_,_,_,_],
    subset(Orders, [1,2,3,4]),
    unique(Orders),

    total(Orders, [1,2,3,4], Li),
    Li=1,
    total(Orders, [1,3,4,2], Wang),
    Wang=1,
    total(Orders, [4,3,1,2], Zhang),
    Zhang=0,
    total(Orders, [4,2,3,1], Yang),
    Yang=2.
```

```
?- getOrders(Orders).
Orders = [3, 2, 4, 1].
```

# 确定选项函数

```
determineAnswer(X,Orders):-
    getOrders(Orders),
    ((X=a, nth0(1,Orders,1,_));
    (X=b, nth0(2,Orders,2,_));
    (X=c, \+(nth0(0,Orders,3,_)));
    (X=d, \+(nth0(3,Orders,4,_)))).

?- determineAnswer(X,Orders).
X = d,
Orders = [3, 2, 4, 1].
```

# 完整程序

```
subset([],_).
subset([X|Y], Z):-
    member(X,Z),
    subset(Y,Z).
unique([]).
unique([X|Y]):-
    not(member(X, Y)),
    unique(Y).
total([], _, 0).
total([A1|A2], [B1|B2], N):-
    (A1=B1, N1=1; A1\=B1, N1=0),
    total(A2, B2, N2),
    N is N1+N2.
```

```
getOrders(Orders):-
    Orders=[_,_,_,_],
    subset(Orders, [1,2,3,4]),
    unique(Orders),
    total(Orders, [1,2,3,4], Li),    Li=1,
    total(Orders, [1,3,4,2], Wang),  Wang=1,
    total(Orders, [4,3,1,2], Zhang), Zhang=0,
    total(Orders, [4,2,3,1], Yang),  Yang=2.
determineAnswer(X,Orders):-
    getOrders(Orders),
    ((X=a, nth0(1,Orders,1,_));
    (X=b, nth0(2,Orders,2,_));
    (X=c, \+(nth0(0,Orders,3,_)));
    (X=d, \+(nth0(3,Orders,4,_)))).
```