AI言語Prologによるプログラム作成のための述語群

吉福　功美

（受理　平成7年5月31日）

On the Predicates Group for the Programming by the AI Language Prolog

Isami YOSHIFUKU

In our laboratory, the application of Prolog language to chemical engineering problems has been studied: the flowgraph making program by Prolog and the continuous system simulation program by Prolog for process control problems.

Taking aim at the important role of predicates in the programming by Prolog, the author has gathered predicates group for many years. In this report, predicate groups added in an appendix are shown; these groups and their relation to programming by Prolog are studied.

緒　言

今まで，BASIC，FORTRAN，PASCALやCのような手続き型言語は数値計算での活用がなされてきたのに対して，PROLOGやLISPは論理型言語であり人工知能言語として数値計算以外の未知の分野での仕事が期待されている。

本研究室ではAI言語Prologの化学工学の諸問題への適用として，非線形代数方程式群で記述されている化学プロセスの設計問題に対するフローグラフ作成プログラム[1]やプロセス制御問題に対する連続システム・シミュレーション・プログラムの作成[2]等の研究を行ってきた。このProlog言語は習得するのに時間がかかるという欠点があるが，それは手続き型に慣れたものが，論理型の言語に戸惑うからである。

本報告は，この言語で述語が大きな役割を持っていることに着目し，長年に渉って，プログラミング時に新しく作成した述語を集めたものである。もともとPrologには組み込み述語というものが数十個含まれているが，実際のプログラミングではこれだけでは不十分であって，少なくとも本報告に記載した述語が必要不可欠であると思われる。なお筆者は文字関係の述語も多数集めているが，本報告ではページ数の関係でこれらは割愛し一般の述語だけを記載することにした。

本報告の述語群は述語についてその定義，内容および使用例の順に並んでいる。またその使用方法については報告AI言語Prologによる線形計画法問題の解法プログラムの開発[3]を参照されたい。

引用文献

1) Yoshifuku, I. and R. Fukumura: J. of Chem. Eng. of Japan, 24, 677 (1991)

2) Yoshifuku, I. et al.: Proceedilngs of Taipei-Kyushu Joint Symposium on Chem. Eng., 243 (1994)

3) 吉福功美：鹿児島大学工学部研究報告，35，105 (1995)

述語群（一般的なもの）

1. add_elm(E,L,R) -> RはリストLの先頭に要素Eを加えたものである。
   add_elm(E,L,[E | L]).
   〈eg〉 L=[1,2,3],E=5のときはR=[5,1,2,3].
2. add_last(L1,A,L2) -> L2はリストL1の最後に要素Aを追加したものである。
   add_last(L1,A,L2):- append(L1,[A],L2).

〈eg〉 L1=[a,b,c], A=dのときはL2=[a,b,c,d].

3. add_lst1(N, L1, L2, L3) -> L3はリストL1のN番目にリストL2を挿入したものである。

```
add_lst1(N, L1, L2, L3) : -
reverse (L2, L22), add_lst 1 a (N, L1, L22, L3).
add_lst 1 a (N, L1, L2, L3) : -
L2=[H 1 | T 1], insert (N, H1, L1, L4),
        add_lst 1 a (N, L4, T1, L3).
add_lst 1 a(N, L1, [], L1).
```

〈eg〉 L1=[1, 3, 5, 6, 8, 10], N=4, L2=[2, 6, 4]のときは
L3=[1, 3, 5, 6, 2, 6, 4, 8, 10].

4. add_lst 2 (L1, L2, L3) -> L3はリストL1の最後にリストL2を挿入したものである。

```
add_lst 2 (L1, L2, L3) : -
L2=[H 1 | T 1], add_last (L1, H1, LL),
        add_lst 2 (LL, T1, L3).
```

〈eg〉 L 1=[75, 1, 49, 2, 10], L2=[23, 10]のときは
L3=[75, 1, 49, 2, 10, 23, 10].

5. add_sid 1 (L1, A, L2) -> L2 はリストL1 の両側に要素Aを付加したものである。

```
add_sid 1 (L1, A, L2) : -
add_elm (A, L1, L11), add_last (L11, A, L2).
```

〈eg〉 リストL1=[3, 5, 6, 8, 10], A=23のときは
L3=[23, 3, 5, 6, 8, 10, 23].

6. add_sid 2 (M1, M2, M3) -> M3 は文字列 M 1の両側に文字 M 2 を付加したものである。

```
add_sid 2 (M1, M2, M3) : -
conv_ 2 (M1, L1), conv_ 2 (M2, L2), L2=[E22],
  add_sid 1 (L1, E22, L3), conv_ 1 (L3, M3).
```

〈eg〉 M1= $ a1b1+c1d1 $, M2= $ + $のときは
M3=+a1b 1 +c1d 1 +.

7. adj_elm (N1, N2, L) -> リストLで2つの要素N 1 , N 2 は隣接している。

```
adj_elm (N1, N2, L) : - append (As, [N1, N2 |
Ys], L).
```

〈eg〉 L=[1, 2, 3, 4]で要素N1=2, N2=3 は隣接している。

8. append (Xs, Ys, Zs) -> リスト Zs は 2 つのリストXs, Ys を連結したリストである。

```
append ([], Ys, Ys).
append ([X | Xs], Ys, [X | Zs]) : -
        append (Xs, Ys, Zs).
```

〈eg〉 Xs=[1, 2, 5, 7], Ys=11, 15, 18]のときは
Zs=[1, 2, 5, 7, 11, 15, 18].

9. betwn_ 1 (I, J, K) -> 整数KがIとJの間にあればtrue (両端を含む)。

```
betwn_ 1 (I, J, I) : - I =〈J.
betwn_ 1 (I, J, K) : - 〈J, I 1 is
I+1, betwn_ 1 (I 1, J, K).
```

〈eg〉 I=2, J=6のときはK=2, 3, 4, 5, 6.

10. betwn_ 2 (I, J, K) -> 整数KがIとJの間にあればtrue (両端を含まない)。

```
betwn_ 2 (I, J, K) : - I〈J, K〈J, K〉I .
```

〈eg〉 I=2, J=6のときはK=3, 4, 5の場合がtrue.

11. betwn_ 3 (L1, L2, L3) -> L3は整数リストL1の要素についてL2=[I, J]間にあるものから成るリストである。

```
betwn_ 3 (L1, L2, L3) : -
    betwn_ 3 (L1, L2, [], L3).
betwn_ 3 (L1, L2, LL, L3) : -
      L2=[I, J], L1=[F | T],
    ifthene | se (betwn_ 2 (I, J, F), LS=
                                [F], LS=[]),
    append (LL, LS, LT),
      betwn_ 3 (T, L2, LT, L3).
betwn_ 3 ([], L2, L3, L3).
```

〈eg〉 L1=[1, 7, 9, 5, 8, 10, 6, 11, 8], L2=[5, 10]のときはL3=[7, 9, 8, 6, 8].

12. betwn_ 4 (L1, L2, E) -> E は整数リスト L 1 の各要素について最初に L2=[I, J]間にあるものである。ないときはE=0 とする。

```
betwn_ 4 (L1, L2, E) : - L2=[I, J], L1=[F | T],
    ifthene | se (betwn_ 2 (I, J, F), E is F,
    betwn_ 4 (T, L2, E).
betwn_ 4 ([], L2, 0).
```

〈eg〉 L1=[1, 9, 5, 10, 6], L2=[5, 10]のときは
E=9.

13. betwn_ 5 (L1, A, L2) -> L2は整数2重リストL 1の要素で整数Aを要素間に含むサブリストである。

```
betwn_ 5 (L1, A, L2) : -
                L1=[H1 | T1], H1=[I, J],
    ifthene | se (betwn_ 2 (I, J, A), L2=[i, j],
    betwn_ 5 (T1, A, L2).
```

〈eg〉 L1=[[1, 5], [5, 9], [9, 13]], A=10のときはL2=[9, 13].

14. conv_ 1(L, M)->整数リストLから文字列Mへの変換。

```
conv 1 (L, M) : - name(A, L), atom_string(A,
M).
```
〈eg〉 L=[65, 47, 40, 84, 115, 43, 66, 41]のときは M=A/(Ts+B).

15. conv_2(M, L) -→ 文字列Mからコードリスト Lへの変換。
```
conv_2(M, L) : - atom_string (A, M),
    name(A, L).
```
〈eg〉 M=A/(Ts+B)のときはL=[65, 47, 40, 84, 115, 43, 66, 41].

16. conv_3 (L1, L2) -→ 整数2重リストL1から2重文字リストL2への変換。
```
conv_3 (L1, L2) : - conv_3 (L1, [], L2).
conv_3 (L1, LL, L2) : - L1=[H1 | T1],
    conv_1(H1,M1), append (LL, [M1],LT),
    conv_3(T1, LT, L2).
conv_3 ([], L2, L2).
```
〈eg〉 L=[[40,97,43,98,41], [40,99,43,100,41]] のときはL2=[(a+b), (c+d)].

17. conv_4(L1, L2) -→ 文字リストL1から2重整数リストL2への変換。
```
conv_4(L1, L2) : - conv_4(L1, [], L2).
conv_4(L1, LL, L2) : - L1=[H1 | T1],
    conv_2(H1, M1), append(LL, [M1], LT),
    conv_4(T1, LT, L2).
conv_4([], L2, L2).
```
〈eg〉 L=[(a+b), (c+b)]のときはL2=[[40, 97, 43, 98, 41], [40, 99, 43, 100, 41]].

18. dbl_dbl 1 (L1, L2, L3) -→ L3は2重リストL1からリストL2を含むという条件つきの2重リストである。
```
dbl_dbl 1 ([[]], L2, [[]]).
dbl_dbl 1 (L1, [], L1).
dbl_dbl 1 (L1, L2, L3) : - asserta (f(mark)),
    member(E1, L1), lst_sub3(L2, E1), X=E1,
    asserta (f(X)), fail.
dbl_dbl 1 (L1, L2, L3) : - collect ([], L3).
```
〈eg〉 L1=[[49, 46], [48, 2], [1, 49, 46, 50], [52, 49, 1]], L2=[49, 46]のときはL3=[[49, 46], [1, 49, 46, 50]].

19. dbl_dbl 2 (L1, L2) -→ L2は2重リストL1から条件付きの2重リストである。L2の要素はL1の要素の両端を除いたもの。
```
dbl_dbl2(L1, L2) : - length(L1, N1),
    dbl_dbl2(L1, L2, N1).
dbl_dbl2(L1, L2, 0).
dbl_dbl2(L1, L2, N) : - L1=[H1 | T1],
    del_sid1(H1, H2),
    exchg_3(L1, 1, H2, L11),
    rotate (1, L11, L12),
    N1 is N- 1,
dbl_dbl 2 (L12, L2, N1).
```
〈eg〉 L1=[[43, 97, 49, 43], [43, 98, 49, 43], [43, 99, 49, 43]]のときはL2=[[97, 49], [98, 49], [99, 49]].

20. dbl_sig 1 (L1, L2) -→ 2重リストL1から各要素の第1要素を集めたリストL2の作成。
```
dbl_sig 1 (L1, L2):- dbl_sig 1 (L1, [], L2).
dbl_sig 1 ([], L2, L2).
dbl_sig 1 (L1, LL, L2) : -
      L1=[H1 | T1], H1=[A, B],
    append (LL, [A], LT),
    dbl_sig 1 (T1, LT, L2).
```
〈eg〉 L1 =[[75, 1], [49, 2], [47, 3], [41, 4], [52, 5], [53, 6]]のときはL2=[75, 49, 47, 41, 52, 53].

21. dbl_sig 2 (L1, L2) -→整数2重リストL1から条件付きリストL2の作成。ただしL1の各要素を43(+ノASCII値) で連結する。
```
dbl_sig 2 (L1, L2) : - dbl_sig 2 (L1, [], L2),
    del_last (L12, L2).
dbl_sig 2 (L1, LL, L2) : - L1=[H1, T],
add_last (H1, 43, L11), append (LL, L11, LT),
dbl_sig 2 (T1, LT, L2).
```
〈eg〉L1=[[107, 49, 165, 115], [107, 50, 165, 115], [108, 49, 165, 115]]のときはL2=[107, 49, 165, 115, 43, 107, 50, 165, 115, 43, 108, 49, 165, 115].

22. dbl_tri 1 (L1, L2, L3) -→ 2個の2重リストL1, L2から3重リストL3の作成。ただしL1, L2の各要素が異なるものを要素する。
```
dbl_tri1(L1, L2, L3):- dbl_tri1(L1, L2, [], L3).
dbl_tri1(L1, L2, LL, L3) : -
                    L1=[H1, T], L2=[H2 | T2],
    ifthenels (H1 ¥== H2, append
                        (LL, [H1, H2]], LT),
    append (LL, [], LT)),
    dbl_tri 1 (T1, T2, L3, L3).
dbl_tri 1 ([], [], L3, L3).
```
〈eg〉 L1=[[1, 20, 35], [40, 97, 20], [15, 26], [25,

14], [18,20]], L2=[[1,20,35], [97,25], [15,26], [25, 15], [17, 20]]のときはL3=[[40, 97, 20], [97, 25]], [[25, 14], [25, 15]], [[18, 20], [17, 20]]].

23. del_blk(L1, L2) -> L2は2重リストL1の最後の要素が[]ならこれを除いたものである。でないならそのままとする。

```
del_blk ([[]], [[]]).
del_blk (L1, L2) :- last (L1, A),
ifthenelse (A=[], del_last (L1, L2), L2=L1).
```

〈eg〉 L1=[[1, 2, 3], [5, 6, 1], []]のときは L2=[[1, 2, 3], [5, 6, 1]]

24. del_elm 1 (X, L, Xs) ->リストXsはリストLの中の要素Xを一つだけ削除した残りのリストである。

```
del_elm 1 (X, [X | Xs], Xs).
del_elm 1 (X, [Y | Ys], [Y | Zs]) :-
        del_elm (X, Ys, Zs).
```

〈eg〉 L=[1,2, 3, 4], X=1のときはXs=[2, 3, 4].

25. del_elm 2 (L1, N1, L2) ->L2はリストL1からN1番目の要素を除去したものである。

```
del_elm 2 (L1, N1, L2) :- length (L1, NN),
    ifthenelse (N1>1,
    rotate (N1-1, L1, L2), L2=L1),
    L2=[H1 | T1], Nr is NN-N1,
    rotate (Nr, T1, L2).
```

〈eg〉 L1=[1, 20, 3, 15, 30, 5, 10, 24], N1=3のときはL2=[1, 20, 15, 30, 5, 10, 24]

26. del_elm 3 (L1, X, L2) ->リストL2はリストL1の中の1個あるいは重複した要素Xを全て削除した残りのリストである。

```
del_elm 3 ([X | LL], X, L2) :-
    del_elm 3 (LL, X, L2).
del_elm 3 ([X | LL], Z, [X | L2]) :- X ¥=Z,
    del_elm 3 (LL, Z, L2).
del_elm 3 ([], X, []).
```

〈eg〉 リストL=[1, 2, 3, 1, 4], X= 1のときは Xs=[2, 3, 4].

27. del_last(L1, L2) -> リストL1から最終の要素を除去したリストL2の作成。

```
del_last (L1, L2) :- length (L1, NN),
    rotate (NN-1, L1, LL), LL=[H1 | L2].
```

〈eg〉 L1=[1, 2, 3, 1, 4]のときはL2=[1, 2, 3, 1].

28. del_lst 1 (L1, L2, L3) ->L3はリストL1から部分リストL2を除去したものである。ただしL2が先頭にあるか，L1の要素が全て異なっている場合。

```
del_lst 1 (L1, L2, L3) :- L2=[H1 | T1],
    select (H1, L1, L4), del_lst 1 (L4, T1, L3).
del_lst 1 (L1, [], L1).
```

〈eg〉 L1=[1, 3, 5, 6, 8, 10, 54, 14, 6], L2=[6, 8, 10, 54]のときはL3=[1, 3, 5, 14, 6].

29. del_lst 2 (L1, N1, L2, N2, LL) -> LLはリストL1から一続きの部分リストL2を除去したものである。

```
del_lst 2 (L1, N1, L2, N2, LL) :-
    num1_lst (L1, L2, Nm), Nr is Nm-1,
    rotate (Nr, L1, L3), del_lst (L3, L2, L4),
    Nt is N1-N2-Nr, rotate (Nt, l4, LL).
del_lst 2 (L1, N1, L2, N2, LL) :-
    write ('not exist).
```

〈eg〉 L 1 =[1, 3, 5, 6, 2, 6, 4, 8, 10, 25], L2=[6, 2, 6, 4]のときはL 3 =[1, 3, 5, 8, 10, 25].

30. del_lst 3 (L1, N1, L2) ->LLはリストL1から先頭のN個の要素を除去したものである。

```
del_lst 3 (L1, 0, M) :- M=L1.
del_lst 3 (L1, N, M) :- L1=[H|T1], N1 is N-1,
    del_lst 3 (T1, N1, M).
```

〈eg〉 L1=[1,3,5,5,10,9,3,4,1], N=6のときは L2=[3, 4, 1].

31. del_lst 4 (L1, L2, LL) ->LLはリストL1からサブリストL2を全て除去したものである。

```
del_lst4 (L1, L2, LL) :- del_lst4 (L1, L2, LL, 0).
del_lst4 (L1, L2, LL, l) :-
    del_lst2 (L1, L2, L3, Nm),
    l1 is l+1, del_lst4 (L3, L2, LL, l1).
```

〈eg〉 L1=[6,2,6,4,8,2,6,4,25,2,6,4,2], L2=[2, 6, 4]のときはLL=[6, 8, 25, 2].

32. del_lst 5 (L1, L2, L3) ->L3は2重リストL1から要素L2を除去したものである。

```
del_lst 5 (L1, L2, L3) :-
    del_lst 5 (L1, L2, [], L3).
del_lst 5 ([], L2, L3, L3).
del_lst 5 (L1, L2, LL, L3) :- L1=[H1 | T1],
    ifthenelse(LS=[], LT=LL,
      append(LL, [LS], LT)),
      del_lst 5 (T1, L2, LT, L3).
```

〈eg〉 L1=[[1, 3, 5], [2, 10], [6, 4, 8, 2]], L2=[2, 10]のときはL3=[[1, 3, 5], [6, 4, 8, 2]].

33. del_lst 6 (L1, L2, L3) -> L 3 は 2 重リストL1からサブ2重リスト L 2 を除去したものである。

```
del_lst 6 (L1, L2, L3) : - L2=[H1 | T1],
  del_lst5 (L1, H1, L31], del_lst6 (L31, T1, L3).
```
〈eg〉 L1=[[1, 3, 5], [2, 10], [6, 4, 8, 2], [5, 1]], L2=[[2, 10], [5, 1]]のときは L3=[[1, 3, 5], [6, 4, 8, 2]]

34. del_sid 1 (L1, L2) -→ L2はリストL1の最初と最後の要素を除去したものである。
```
dil_sed 1 (L1, L2):- L1=[H1 | T1],
    dil_sed 1 (T1, [], L2).
del_sid 1 (L1, LL, L2) : -
                    L1=[H1 | T1], LS=[H1],
  append (LL, LS, LT), del_sid1 (T1, LT, L2).
del_sid 1 ([A], L2, L2).
```
〈eg〉 リストL1=[1, 2, 3, 4, 5]のときは L2=[2, 3, 4].

35. dif_lst (L1, L2) -→ 整数リストL1の各要素の差を要素とするリストL2の作成。
```
dif_lst (L1, L2) : - length(L1, N),
  dif_lst (N, L1, [], L2).
dif_lst (1, _, L2, L2).
dif_lst (N, L1, LL, L2) : -
                L1=[H1 | T1], T1=[H2 | T2],
  A is H2-H1, N1 is N-1, append (LL, [A], LT),
  dif_lst (N1, T1, LT, L2).
```
〈eg〉 L1=[1, 12, 28, 35, 54], N=5のときは L2=[11, 16, 7, 19].

36. exchg_1 (L1, N, L2) -→ リストL2はリストL1のN番目の要素について前後を交換したものである。
```
exchg_1 (L1, N, L2) : -
  sampling (N, L1, A), N1 is N-1,
  lst_sub 1 (L1, N1, L3, LL),
                L3=[H | T1], T2=[A | LL],
  append (T1, T2, L2).
```
〈eg〉 L1=[2, 8, 10, 6, 25, 49, 43], N=4のときはその要素は6で,
L2=[25, 49, 43, 6, 2, 8, 10].

37. exchg_2 (L1, N, A, L2) -→ L2はリストL1の第N番目の要素をリストAと交換したものである。
```
exchg_2 (L1, N, A, L2) : -
  del_elm 2 (L1, N, L11),
  ins_elm 1 (L11, N, A, L12), flatten (L12, L2).
```
〈eg〉 L1=[1, 7, 9, 5, 8, 10, 6, 11, 8], N=2, A=[9, 9, 9] のときは L2=[1, 9, 9, 9, 9, 5, 8, 10, 6, 11, 8].

38. exchg_3 (L1, L2, LL) -→ 要素が重複したリストL1から2重リストL2を用いてリストLLを作成する。
```
exchg_3 (L1, L2, LL) : -
  length (L1, N), L2=[L21 | [A1]],
  length (L21, N2),
  exchg_3 b (L1, L21, A, N21, LL, N1).
exchg_3 b (LL, L21, A, N21, LL, 0).
exchg_3 b (LL, L21, A, N21, LL, 1) : -
                        L1=[H1 | T1],
  ifthenelse (H1=A,
                exchg_3 b (T1, L21, N21, L3),
  rotate (1, L1, L3), l1 is 1-1,
  exchg_3 b (L3, L21, A, N21, LL, l1).
exchg_3 b (T1, L21, NL1, L3) : -
  append(L21, T1, L31), rotate(NL1, L31, L3).
```
〈eg〉 L1=[1, 5, 7, 9, 5, 8], L2=[[1, 2, 3], [5]] のときはLL=[1, 1, 2, 3, 7, 9, 1, 2, 3, 8].

39. exchg_4 (L1, L2, LL) -→ 要素が重複したリストL1から2重リストL2を用いてリストLLを作成する。
```
exchg_4 (L1, L2, LL) : - L2=[B, A],
  length (L1, N1), length (A, NA),
  exchg_4 c (L1, B, A, NB, LL, N1).
exchg_4 c (LL, B, A, NB, LL, 0).
exchg_4 c (L1, B, A, NB, LL, 1) : -
                        L1=[H1 | T1],
  length (A, NA), lst_div 5 (L1, NA, H1, T2),
  ifthenelse (H1=A, exchg_4c(T2, B, NB, L3),
    l1 is 1-NA, (rotate (1, L1, L3), l1 is 1-1)),
    exchg_4 c (L3, B, A, NB, LL, l1).
exchg_4 c (T2, B, NL1, L3) : -
  append (B, T2, L31), rotate(NL1, L31, L3).
```
〈eg〉 L1=[98, 165, 122, 49, 110, 43, 100, 122, 49, 110, 47, 100, 116], L2=[[120, 49], [122, 49, 110]] のときは LL=[98, 165, 120, 49, 43, 100, 120, 49, 47, 100, 116].

40. exchg_5 (L1, L2, LL) -→ 要素が重複したリストL1から3重リストL2を用いてリストLLを作る。
```
exchg_5 (L1, L2, L3, LL) : - L2=[H1 | T1],
  exchg_3 (L1, H1, LL1), exchg_5 (LL1, T1, LL).
```
〈eg〉 L1=[98, 165, 100, 121, 47, 100, 116, 43, 122, 61, 97, 165, 120], L2=[[[120, 49], [120]], [[121,

50], [121]], [[107, 49], [97]], [[107, 50], [98]]]
のときはLL＝[107, 50, 165, 100, 121, 50, 47, 100, 116, 43, 122, 61, 107, 49, 165, 120, 49]

41. exchg_6(L1, L2, LL) -> 要素が重複したリストL1から3重リストL2を用いてリストLLを作る。
exchg_6(L1, L2, LL) :- L2=[H1 | T1],
exchg_4(L1, H1, LL1), exchg_6(LL1, T1, LL).
〈eg〉 L1＝[3, 6, 1, 5, 7, 9, 5, 7, 8, 3, 6, 10, 5, 7, 11, 9, 8, 5], L2=[[[1, 2, 3], [5, 7]], [[50, 40], [3, 6]], [[10, 20, 30], [8, 5]]]のときはLL=[50, 40, 1, 1, 2, 3, 9, 1, 2, 3, 8, 50, 40, 10, 1, 2, 3, 11, 10, 20, 30]

42. flatten(Xs, Ys) -> YsはリストXsを平坦化したものである。
flatten([X | Xs], Ys3) :- flatten(X, Ys1),
flatten(Xs, Ys2), append(Ys1, Ys2, Ys3).
flatten(X, [X]) :- atomic(X), X ¥=[].
flatten([], []).
〈eg〉 Xs=[h, [i, j], a, [b, [c, d], e], [f, g]] のときはYs=[h, i, j, a, b, c, d, e, f, g]

43. fnd_lst(L1, L2, N) -> NはリストL1の先頭から何個ヘッドを取り除いたらリストL2を含むサブリストが現れるかその個数である。
fnd_lst(L1, L2, Nm) :-
fnd_lst(L1, L2, 1, 1, Nn), Nm is Nn-1.
fnd_lst(L1, L2, 0, _, Nm).
fnd_lst(L1, L2, 1, Nr, Nm) :-
iftthenelse(append(L2, L3, L1), No=0, No=1),
L1=[H1 | T1], Ns is Nr+1,
ifthenelse(No=1, fnd_lst(T1, L2, No, Ns, Nm),
Nm is Nm-1).
〈eg〉 L1＝[1, 3, 5, 6, 2, 6, 4, 8, 10], L2＝[2, 6, 4, 8]のときはNm＝4.

44. func_1(L1, L2) -> L2は整数リストL1の各要素にたいしてある関数を適用した新しいリストである。
func_1(L1, L2) :- func_1(L1, L2, []).
func_1([A | R], H, T):-
func(A, H, M), func_1(R, M, T).
func_1(A, [B | D], D) :- func(A, B).
func(A, B) :- B is 2*A.
〈eg〉 リストL1=[1, 2, 3, 4, 5]各要素を2倍したリストはL2＝[2, 4, 6, 8, 10].

45. ins_elm1(N, E, L1, L2) -> L2はリストL1Zの先頭からN番目に要素Eを挿入したものである。
ins_elm1(0, E, L, [E | L]) :- !.
ins_elm1(_, E, [], [E]) :- !.
ins_elm1(N, E, [H|T], [H | R]) :- N1 is N-1,
ins_elm1(N1, E, T, R).
〈eg〉 L1＝[1, 3, 5, 8, 4], N＝3, E=10のときはL2＝[1, 3, 5, 10, 8, 4].

46. ins_elm2(L1, A, L2) -> L2はリストL1のすべての要素間に要素Aを挿入したものである。
ins_elm2(L1, A, L2) :- length(L1, N2), NL is N2+N2-1, ins_elm2(NL, 1, L1, , A, , L2).
ins_elm2(NL, N, L2, A, L2) :- NL 〈= N.
ins_elm2(NL, N, L1, A, L2) :-
ins_elm1(N, A, L1, L11), N1 is N+2,
ins_elm2(NL, N1, L11, A, L2).
〈eg〉 L1=[5, 8, 4, 8, 9], A=3のときはL2=[5, 3, 8, 3, 4, 3, 8, 3, 9].

47. ins_last(A, L1, L2) -> L2はリストL1の最後に要素Aを挿入したものである。
ins_last(X, [], [X]).
ins_last(X, [F | R], [F | S]) :-
ins_last(X, R, S).
〈eg〉 L1=[a,b,c], A=dのときはL2=[a,b,c,d].

48. ins_lst(L1, L2, A, L3) -> L3はリストL1に対して番号リストL2の番号のところに要素Aを挿入したものである。
ins_lst(L1,L2,A,L3) :- L2=[N|T], N1 is N-1,
ins_elm1(N1, A, L1, LL),
ins_lst(LL, T, A, L3).
ins_lst(L3, [], A, L3).
〈eg〉 L1=[10, 20, 30, 40, 50], L2=[1, 4, 9], A=99のときはL3=[99, 10, 20, 99, 30, 40, 50, 60, 99]

49. intsect(L1, L2, L3) -> L3は2個のリストL1, L2の共通要素からなるリスト
intsect(L1, L2, L3) :- intsect(L1, L2, [], L3).
intsect([], L2, L3, L3).
intsect(L1, L2, LL, L3) :- L1=[H1 | T1],
ifthenslse
(member(H1, L2), LS=[H1],LS=[]),
append(LL, LS, LT),
intsect(T1, L2, LT, L3).
〈eg〉 L1=[a, b, c, d], L2=[e, f, c, g, d]のときはL3＝[c, d].

50. last(L, X) -> Xはリストの最終要素である。

```
last([X], X).
last([F | R], X) :- last(R, X).
```
〈eg〉 L=[a, b, c]のときはX=c.

51. lst_div 1 (L1, L2, LL1, LL2) -> LL1, LL2 はリストL1を区間リストL2で大小関係について分割した2個のリストである。

```
lst_div 1 (L1, L2, LL1, LL2) :-
lst_div 1 (L1, L2, [], [], LL1, LL2).
lst_div 1 (L1, L2, L01, LL1, LL2) :-
L1=[H | T], L2=[l1, l2],
  ifthenelse((H>l1, H<l2), LS=[H], LT=[H]),
  ifthenelse(LS=[], L11=L01,
  append(L01, LS, L11)),
  ifthenelse(LY=[], L12=L02,
  append(L02, LT, L12)),
  lst_div 1 (T, L2, L11, L12, LL1, LL2).
```
〈eg〉 L 1 =[1, 3, 8, 6, 13, 5, 11, 10, 16, 22, 12], L 2 =[5, 12]のときは
LL1=[8,6,11,10], LL2=[1,3,13,5, 16, 22, 12].

52. lst_div 2 (L1, N, LL1, LL2) -> LL 1, LL 2 は整数リストL 1 をある数Nで大小関係について分割したものである。

```
lst_div 1 ([X | Xs], N, [X | Ls], Bs) :-X=<Y,
  lst_div 2 (Xs, N, Ls, Bs).
lst_div 1 ([X | Xs], N, Ls, [X | Bs]) :-X>N,
  lst_div 2 (Xs, N, Ls, Bs).
lst_div 2 ([], N, [], []).
```
〈eg〉 L 1 =[1, 12, 35, 3, 5, 6, 8, 10, 26], N=10 のときはLL1=[1, 3, 5, 6, 8, 10], LL2=[12, 35, 26].

53. lst_div 3 (L1, A, L2, L3) -> L2, L3はリストL 1 をある要素Aでその前後に分割したものである。

```
lst_div 3 (L1,A,L2,L3) :-lst_num(L1, A, NA),
  N1 is NA-1, lst_sub 1 (L1, N1, L33, L2),
  L33=[H | L3].
```
〈eg〉 L1=[1, 12, 35, 3, 38, 5, 6, 8, 10, 26], A=38 のときはL2=[1, 12, 35, 3], L3=[5, 6, 8, 10, 26].

54. lst_div 4 (L1, A, B, LL) -> LLはリストL1の2個の要素A, B間にある要素のリストである。

```
lst_div 4 (L1, A, B, LL) :-
  lst_div 3 (L1, A, L2, L3),
  lst_div 3 (L3, B, LL, L5).
```
〈eg〉 L1=[100, 122, 49, 110, 47, 100, 116, 61, 40, 114], A=100, B=47のときはLL=[122, 49, 110].

55. lst_div 5 (L1, N, L2, L3) -> L2 はリストL 1 の先頭からN個取った要素のリストである。L 3 は残りのリスト。

```
lst_div5(L1, N, L2, L3) :-div5(L1, N, [], L2),
  append (L2, L3, L1).
div 5 (L1, 0, L2, L2).
div 5 (L1, N, LL, L2) :- L1=[H1 | T1],
append (LL, [H1], LS), N1 is N-1,
  div 5 (T1, N1, LS, L2).
```
〈eg〉 L1=[20,4,3,1,2,3,4, 5, 6, 7], N=4のときは L2=[20, 4, 3, 1], L3=[2, 3, 4, 5, 6, 7].

56. lst_div 6 (L1, A, L2) -> リストL 1 で，ある要素が重複しているときその要素で場所について分割した2重リストL 2 を作る。

```
lst_div 6 (L1, A, L2) :- length (L1, NO),
lst_mk 6a (L1, A, L11), N11 is NO+1,
add_last (L11, N11, L12), div 5 (L1, N, [], L2),
append (L2, L3, L1), L13=[0 | L12],
  dif_lst (L13, L14), LL1=[0 | L1],
  div 6 (LL1, L14, L2).
div 6 (L1, LL, LS, L2) :-
                L1=[L0 | LL1], LL=[H1 | T1],
lst_div 5 (LL1, H1, L14, L15),
  append (LS, [L14], LT, L2).
```
〈eg〉 L 1=[1, 12, 35, 3, 38, 5, 6, 8, 10, 26, 38, 12, 45, 38, 10], A=38, のときは L 2 =[[1, 12, 35, 3], [5, 6, 8, 10, 26], [12, 45], [10]].

57. lst_num (L1, A, NA) -> NAはリストL 1 の中で要素Aの番号である。

```
lst_num(L1,A,NA) :- lst_num(L1, 1, A, NA).
lst_num(L1, N, A, NA) :- L1=[Y | T],
  ifthenelse(N=Y, NA is N, (N1 is N+1,
  lst_num(T, N1, A, NA))).
lst_num([], N, A, 0).
```
〈eg〉 L1=[40, 84, 6, 70, 115, 43, 66, 41, 43, 65], A=115のときはNA=5.

58. lst_sub 1 (L1, N, L3, L2) -> L2, L3 はリストL 1 の先頭のN個の要素からなるリストと残りのリストである。

```
lst_sub 1 (L1, N, L2, L3) :-
del_lst 3 (L1, N, L2), append(L3, L2, L1).
```
〈eg〉 L1=[1,3, 5, 5, 10, 9, 3, 4, 1], N=3のときは L2=[1, 3, 5], L3=[5, 10, 9, 3, 4, 1]

59. lst_sub 2 (L1, N1, N2, LL) -> LLはリストL1のN1番からN2番目までの要素からなるリストで

ある。

```
lst_sub 2 (L1, N1, N2, LL) : - N11 is N1-1,
lst_sub 1 (L1, N11, L2),  N3 is N2-N11,
  lst_sub 1 (L2, N3, LL).
```

〈eg〉 L1=[1, 3, 5, 5, 10, 9, 3, 4, 1], N1=2, N2=3 のときは LL=[3, 5]

60. lst_sub 3 (L1, L2) -→ リストL2 がリストL1 のサブリストであるか否か。

```
lst_sub 3 (L1, L2) : - prefix(LS, L2),
  suffix (L1, LS).
```

〈eg〉 L2=[3, 8, 9]はL1=[1, 14, 6, 12, 22, 3, 8, 9, 15]のサブリストである。

61. lst_sub 4 (L1, N, R, L2) -→ L2 はリストL1 をR回回転して先頭からN個の要素のリストである。

```
lst_sub 4 (L1, N, R, L2) : -
  lst_sub 1 (L1, N, _, L11),
  rotate (R, L1, L12), lst_sub1 (L12, N, _, L2).
```

〈eg〉 L1=[10, 20, 30, 40, 50, 60], R=3, N=4 のときは L2=[40, 50, 60, 10].

62. map_01(L1, L2) -→ L2 は整数リストL1 を写像したリストである。

写像は g(X)なら 1, でないなら 0 とする。

```
map_01 ([], []).
map_01 ([H | T], [V | R]) : - f (H, V),
  map_01 (T, R).
f (X, 1) : - g(X), !.
f (X, 0).
```

〈eg〉 L1=[97, 43, 47, 40, 98, 49, 43], g(X)をX=43なら 1, でないなら0とする。このとき L2=[0, 1, 0, 0, 0, 0, 1]

63. max_num (L1, Max, No) -→ リストL1の要素で最大のものMaxとその番号Noを求める。

```
max_num (L1, Max, No) : -
max 2 (L1, Max), member (Max, L1, No).
max 2 ([], error).
max 2 ([X], X) : - !.
max 2 ([H | L], Max) : -
max 2 (L, M), (H〉M, Max is H;Max is M), !.
```

〈eg〉 L 1 =[1, 3, 5, 6, 8, 10, 5, 6]のときはmax=10, No=6.

64. member (E, L) -→ Eはリスト Lの要素である。

```
member (X, [X | T]).
member (X, [Y | T]) : - member (X, T).
```

〈eg〉 E= 1 はリストL1=[2, 1, 3]の要素である。

65. mk_lst 1 (L1, N, L2) -→ L2 はリストL1 の要素でNより大きいものから成るリストである。

```
mk_lst 1 (L1,N,L2) : - mk_lst 1 (L1,N, [], L2).
mk_lst 1 ([], N, L2, L2).
mk_lst 1 (L1, L2, LL, L3) : - L1=[H1 | T1],
  ifthenslse (H1〉N, LS=[H1], LS=([]),
  append (LL,LS,LT), mk_lst 1 (T1,N,LT,L2).
```

〈eg〉 L 1 =[1, 14, 6, 12, 22, 3, 8, 9, 15], N=10 のときは L 2 =[14, 12, 22, 15].

66. mk_lst 2 (L1, N1, N2, L2) -→ L2 は 2 重リスト L1 のN1 番目からN2 までの第 1 要素からなるリストである。

```
mk_lst 2 (L1, N1, N2, L2) : -
mk_lst 2 (L1, N1, N2, , [], L2).
mk_lst 2 ([], N1, N2, L2, L2).
mk_lst 2 (L1,N1,N2,LL,L2) : -L1=[H1 | T1],
  H1=[NA,l1], ifthenslse ((l1〉=N1,l1=〈N2),
  LS=[NA], LS[]), append (LL, LS, LT),
  mk_lst 2 (T1, N1, N2, LT, L2).
```

〈eg〉 2 重 リ ス ト L1=[[75,1], [49, 2], [47, 3], [41, 4], [52, 5], [53, 6]], N1=2, N2=5のときは L2=[49, 47, 41, 52].

67. mk_lst 3 (L1, NA, L2) -→ L2 は2重リストL1の第 1 要素が NAであるものの第ふ要素からなるリストである。

```
mk_lst 3 (L1, NA, L2) : -
  mk_lst 3 (L1, NA, [], L2).
mk_lst 3 ([], NA, L2, L2).
mk_lst 3 (L1, NA, LL, L2) : -
                    L1=[H1 | T1], H1=[l1, lB],
  append (LL, LS, LT),
  mk_lst 3 (T1, NA, LT, L2).
```

〈eg〉 L1=[[75, 1], [49, 2], [47, 3], [49, 4], [52, 5], [49, 6]], NA=49のときはL2=[2, 4, 6].

68. mk_lst 4 (L1, L2, L3) -→ L3 は整数リストL1 の要素でL2の区間にある要素からなるリストである。

```
mk_lst 4 (L1, L2, L3) : -
  mk_lst 4 (L1, L2, [], L3)
mk_lst 4 ([], L2, L3, L3).
mk_lst 4 (L1, L2, LL, L3) : -
                    L1=[H1 | T1], L2=[l1, l2],
  ifthenelse ((H1〉l1,H1〈l2),LS=[H1],LS=[]),
  append (LL,LS,LT), mk_lst 4 (T1,L2,LT, L3).
```

〈eg〉 L1 =[3, 10, 12, 14, 15, 17, 18, 19, 20], L2=

[12, 17]のときはL3＝[14, 15].

69. mk_lst 5 (L1, L2, L3) -→ L3は同じ個数の2個のリストL1, L2から要素の差異を表すリストである。

```
mk_lst 5 (L1, L2, L3) : -
  mk_lst 5 (L1, L2, [], L3)
mk_lst 5 ([], L2, L3, L3).
mk_lst 5 (L1, L2, LL, L3) : -
                L1=[H1 | T1], L2=[H2 | T2],
  l is H1-H2, LS=[l], LS=[]),
  append (LL, LS, LT),
  mk_lst 5 (T1, T2, LT, L3).
```

〈eg〉 L 1 ＝[1, 3, 4, 6, 7], L2 ＝[1, 2, 4, 5, 8]のときはL3 ＝[0, 1, 0, 1, -1].

70. mk_lst 6 (L1, A, L2) -→ L2は重複した要素を持つリストL1について，ある重複要素Aの番号リストである。

```
mk_lst 6 (L1, A, L2) : -
sig_dbl 1 (L1, LL), mk_lst 3 (LL, A, L2).
```

〈eg〉 L1＝[1, 2, 3, 5, 1, 2, 3, 2, 4, 2, 2], A＝2のときはL2＝[2, 6, 8, 10, 11]

71. mk_lst 7 (L1, N, L2) -→ L2は2重リストL1から条件つきのリストである。条件はL2の要素＝L1の第1要素＋(番号-1)＊N

```
mk_lst 7 (L1, N, L2) : -
L 0 ＝L1, mk_lst 7 (L0, L1, N, [], L2).
mk_lst 7 (L0, [], N, L2, L2).
mk_lst 7 (L0,L1, N, LL, L2) : - L1=[H1 | T1],
  lst_num (L0, H1, Nm),
  A is l1+(Nm-1)*N,  LS=[A],
  append (LL, LS, LT),
  mk_lst 7 (L0, T1, N, LT, L2).
```

〈eg〉 L1=[[1, 1], [3, 2], [6, 3], [9, 4]], N＝2のときはL2 ＝[1, 5, 10, 15].

72. mk_lst 8 (L1, L2, L3) -→ L3はリストL1について2重リストL2の区間ある要素からなるリストである。

```
mk_lst 8 (L1, L2, L3) : -
  mk_lst 8 (L1, L2, [], L3).
mk_lst 8 (L1, [], L3, L3).
mk_lst 8 (L1, L2, LL, L3) : - L2=[H2 | T2],
  lst_div 1 (L1, H2, LS, LSS),
  append (LL, LS, LT),
  mk_lst 8 (L1, T2, LT, L3).
```

〈eg〉 L1 ＝[6, 13, 16, 22, 26, 28, 3040], L2 ＝[[1, 15], [17, 25], [26, 31]]のときはL3=[6, 13, 22, 28, 30].

73. no_dobl (Xs, Ys) -→ リストYsはリストXsから重複した要素を除いたリストである。

```
no_dobl ([X | Xs], Ys) : -
  member (X〈Xs), no_dobl (Xs, Ys).
no_dobl ([X | Xs], [X | Ys]) : -
  nonmember (X, Xs), no_dobl(Xs, Ys).
no_dobl ([], []).
```

〈eg〉 Xs=[a, b, c, b]のときはYs=[a, b, c].

74. num_lst (N1, N2, L1) -→ L1は2個の整数N1, N2間の整数を要素とするリストである。

```
num_lst (N1, N2, LL) : - N3 is N2+1,
  num_lst ([], N1, LL, N3).
num_lst (LL, NN, LL, NN).
num_lst (LL, N1, LS, NN) : -
  append(LL, [N1], L2),
  N2 is N1+1, num_lst (L2, N2, LS, NN).
```

〈eg〉 N1=55, N2=62のときは
L1 ＝[55, 56, 57, 58, 59, 60, 61, 62].

75. ordered (X) -→ 整数リストXは昇順に順序づけられているか。

```
ordered ([X]).
ordered ([X, Y | Ys]) : -
  X=〈Y, ordered ([Y | Ys]).
```

〈eg〉 L1 ＝[1, 3, 5, 8, 9]は昇順である。

76. pair_lst (L1, L2, L3) -→ L3は2個のリストL1, L2から各要素のペアを要素とする2重リストである。

```
pair_lst (L1, L2, L3) : -
pair_lst (L1, L2, [], GG1), reverse (GG1, G1).
pair_lst ([], [], G1, G1).
pair_lst (L1, L2, B, GG) : -
L1=[H1 | T1], L2=[H2 | T2], E1=[H1, H2],
  car_cdr (G1,E1,B), pair_lst (T1,T2,G1,GG).
```

〈eg〉 L1＝[0, 1, 2, 3], L2＝[6, 7, 8, 9]のときはL3=[[0, 6], [1, 7], [2, 8], [3, 9]].

77. prefix (L1, LL) -→ リストL1はリストLLの接頭部である。

```
prefix ([], Ys).
prefix ([X | Xs], [X | Ys]) : -
  prefix (Xs, Ys).
```

〈eg〉 L1=[a, b]はリストLL＝[a, b, c]の接頭部である。

78. prep_1 (L1, A1, A2) -→ A2はリストL1である

要素A1の一つ手前の要素である。A1がL1の先頭要素の時はA2＝0とする。

```
prep_1(L1, A1, A2) :- lst_num(L1, A1, N1),
  NN is N1-1, sampling(NN, L1, A2).
prep_1(L1, 1, 0).
```

〈eg〉 L1＝[1, 4, 11, 15, 20, 25, A1＝20のときはA2＝15.

79. prt_n(L, N)->リストLの要素を先頭からN個書き出す。

```
prt_n(L, N) :- ctr_set(0, 0),
  NN is N-1, repeat, ctr_inc(0, C),
  [! rotate(C, L, L2), L2 =[H1, T1],
  write('H1='), write(H1), nl !], c=:=NN.
```

〈eg〉 L1＝[a, b, c, d, e], N＝3のときはa,b,c.

80. prt_lst(L)->リストLの要素を先頭から全部書き出す。

```
prt_lst([]).
prt_lst(L) :- L=[H | T],
  write(H), nl, prt_lst(T).
```

〈eg〉 L=[1, 2, 3, 4]のときは1 2 3 4.

81. q_sort(L1, L2)->L2は整数リストL1をソートしたものである。

```
q_sort(L1, L2) :- q_sort(L1, L2, []).
q_sort([F | R], L2, M) :- split(F, R, L, G),
  q_sort(L, L2, [F | U]), q_sort(G, U, M).
q_sort([], X, X).
split(_, [], []).
split(N, [F | R], [F | L1], G) :-
  N>F<split(N, R, L1, G).
split(N, [F | R], L, [F | G1]) :-
  N<F, split(N, R, L, G1).
```

〈eg〉 L1＝[4, 3, 6, 1, 2, 5]のときはL2＝[1, 2, 3, 4, 5, 6].

82. range_1(M, N, Ns)->Nsは整数MとNを含む区間の整理のリストである。

```
range_1(M, N, [M | Ns]) :-
  M<N, M1 is M+1,
  range_1(M1, N, Ns).
range_1(N, N, [N]).
```

〈eg〉 M＝55, N＝62間のときはNs＝[55, 56, 57, 58, 59, 60, 61, 62].

83. repla_1(L1, N, A, L2)->L2はリストL1の第N番目の要素を他の要素Aで置き換えたものである。

```
repla_1([C | L], 1, A, [A | L]).
repla_1([C| L], N, A, [C | LL]) :- N1 is N-1,
  repla_1(L, N1, A, LL).
```

〈eg〉 L1＝[a, b, c, d, e], N＝4, A＝kのときはL2＝[a, b, c, k, e].

84. repla_2(L1, A, S, L2)->L2はリストL1の要素Aを他の要素Sで置き換えたものである。

```
repla_2(L1, A, S, L2) :- listn(L1,1,A,MA),
repla_2(L1, MA, S, L2).
```

〈eg〉 L1＝[40, 84, 6, 70, 115, 43, 68], A＝115,B＝68のときはL2＝[40, 84, 6, 70, 68, 43, 68].

85. reverse(L1, L2)->リストZsはリストLの反転リストである。

```
reverse([], []).
reverse([X | Xs], L2) :- reverse(Xs, Ys),
  append(Ys, [X], L2).
```

〈eg〉 L1＝[1, 3, 8, 10, 25, 4, 6]のときはL2＝[6, 4, 25, 10, 8, 3, 1]

86. rotate(N, L1, L2)->L2はリストL1をN回転したものである。

```
rotate(0, L1, L1).
rotate(N, L1, L2) :- L1=[H | T], N1 is N-1,
  append(T, [H], LL), rotate(N1, LL, L2).
```

〈eg〉 L1＝[1, 2, 3, 4, 5], N＝2のときはL2＝[3, 4, 5, 1, 2].

87. same_lst(L1, L2, No)->2個のリストL1, L2が同じ要素を持つならNo＝0, 異なるならNo＝1.

```
same_lst(L1, L2, No) :- mk_lst3(L1,L2,L3),
  ifthenelse(member(1, L3), No=1, No=0).
```

〈eg〉 L1＝[1, 3, 5, 3, 4, 1], L2＝[0, 3, 5, 3, 4, 1]のときはNo＝1.

88. sampling(N, L1, A)->AはリストL1のN番目の要素である。

```
sampling(1, L, M) :- L=[Lx | Ly].M=Lx.
sampling(Nb, L, M) :- N is Nb-1,
  L=[Lx | Ly], sampling(N, Ly, M).
```

〈eg〉 L1＝[a, b, c, d], N＝3のときはA＝c.

89. sig_dbl1(L1, L2)->整数リストL1から番号2重リストL2の作成。

```
sig_dbl1(L1, L2) :- sig_dbl1(L1, 0, [], L2).
sig_dbl1([], N, L2, L2).
sig_dbl1(L1, N, LL, L2) :- L1=[H1 | T1],
  N1 is N+1, LS=[H1, N1],
  append(LL, [LS], LT),
  sig_dbl1(T1, N1, LT, L2).
```

〈eg〉 L1=[75, 49, 47, 41, 52]のときは
L2=[75, 1], [49, 2], [47, 3], [41, 4], [52, 5]].

90. sig_dbl 2 (L1, L2) -→整数リストL1から閉区間2重リストL2の作成。
```
sig_dbl 2 (L1, L2) : - sig_dbl 2 (L1, [], L2).
sig_dbl 2 ([T2], L2, L2).
sig_dbl 2 ([H1|T1], LL, L2) : - T1=[H2 | T2],
  LS=[H1, H2], append (LL, [LS], LT),
  sig_dbl 2 (T1, LT, L2).
```
〈eg〉 L1=[1, 3, 5, 8, 12, 20]のときは
L2=[[1, 3], [3, 5], [5, 8], [8, 12], [12, 20]].

91. sig_dbl 3 (L1, L2) -→偶数個の要素の整数リストL1から開区間2重リストL2の作成。
```
sig_dbl 3 (L1, L2) : - sig_dbl 3 (L1, [], L2).
sig_dbl 3 ([], L2, L2).
sig_dbl 3 ([H1 | T1], LL, L2) : -
                    T1=[H2 | T2], LS=[H1, H2],
  append (LL, [LS], LT),
  sig_dbl 3 (T2, LT, L2).
```
〈eg〉 L1=[1, 3, 5, 8, 12, 20]のときは
L2=[[1, 3], [5, 8], [12, 20].

92. sig_dbl 4 (L1, L2, L3) -→L3は整数リストL1から2重リストL2を区間として参照した2重リストである。
```
sig_dbl 4 (L1, L2, L3) : -
  sig_dbl 4 (L1, L2, [], L3).
sig_dbl 4 (L1, L2, LL, L3).
sig_dbl 4 (L1, L2, LL, L3) : -
  L2=[H1 | T1], H1=[N1 | N1],
  lst_sub 2 (L1, N11, N2, LS),
  append (LL, [LS], LT),
  sig_dbl 4 (L1, T1, LT, L3).
sig_dbl 4 (L1, [], L3, L3).
```
〈eg〉 L1=[40, 97, 49, 43, 41, 40, 98, 49, 43, 41, 50, 40, 52, 41], L2=[[1, 5], [6, 10], [12, 14]]のときはL3=[[40, 97, 49, 43, 41], [40, 98, 49, 43, 41], [40, 52, 41]].

93. sig_dbl 5 (L1, L2) -→L2は文字リストL1の各要素を整数にした2重リストである。
```
sig_dbl 5 (L1, L2) : - sig_dbl 5 (L1, [], L2).
sig_dbl 5 (L1, LL, L2) : - L1=[H1 | T1],
  conv_2 (H1, L11), append (LL, [LL1], LT),
  sig_dbl 5 (T1, L2, LT).
sig_dbl 5 ([], L2, L2).
```
〈eg〉 L1=[k1k2, k1(k2+k3)]のときは
L2=[[107,49,165,107,50], [107,49,165,40,107,50, 43, 107, 51, 41]].

94. sig_dbl 6 (L1, L2, L3) -→L3は2つのリストL1とL2の要素の対応が異なる時そのペアを要素とする2重リストである。
```
sig_dbl 6 (L1, L2, L3) : -
  sig_dbl 6 (L1, L2, [], L3).
sig_dbl 6 (L1, L2, LL, L3) : -
  L1=[H1 | T1], L2=[H2 | T2],
  ifthenelse (H1 ¥==H2,
  append (LL, [[H1, H2]], LT),
  append (LL, [], LT)),
  sig_dbl 6 (T1, T2, LT, L3).
sig_dbl 5 ([], [], L3, L3).
```
〈eg〉 L1=[12, 20, 40, 97, 20, 15, 26], L2=[1, 20, 35,97,25,15,25]のときはL3=[[12,1], 40,35], [20, 25], [26, 25]].

95. sub_lst 1 (Sub, L) -→リストSubはリストLの部分リストである。
```
sub_lst (Xs, Ys) : - prefix(Ps, Ys),
  suffix (Xs, Ps).
```
〈eg〉リストSub=[3, 4, 5]はリストL=[1, 2, 3, 4, 5, 6, 7, 8]の部分リストである。

96. sub_lst 2 (L1, R, N, L2) -→L2はリストL1をR回回転し前からN個とったものである。
```
sub_lst 2 (L1, R, N, L2) : -
  rotate (R, L1, L12), lst_sub 1 (L12, N, _, L2).
```
〈eg〉 リストL1=[5,10, 20, 23, 30, 40, 50, 55, 60], R=5, N=6のときはL2=[40, 50, 55, 60, 5, 10]である。

97. suffix (Suf, L) -→リストSufはリストLの接尾部である。
```
suffix (Xs, Xs).
suffix (Xs, [Y | Ys]) : - suffix (Xs, Ys).
```
〈eg〉 Suf=[b, c]はリストL=[a, b, c]の接尾部である。

98. sum_lst (L, Sum) -→Sumは整数リストLの要素の和である。
```
sum_lst (ls, Sum) : - sum_lst (ls, 0, Sum).
sum_lst ([1|ls],Temp,Sum) : - T1 is Temp+1,
  sum_lst (ls, T1, Sum).
sum_lst ([], Sum, Sum).
```
〈eg〉 L1 =[1, 2, 3, 4]のときはSum=10.