

理解度テスト合格者は応用プログラムコース (2-3 ページ) または基本プログラムコース (1 ページ) にすすみ, 不合格者 (欠席者を含む) は基本プログラムコース (1 ページ) にすすむ。

理解度テストに合格しなければ単位は取得できないが, テストに合格しても単位取得を確約するものではない。また, 最終回に再度コース別理解度テストを行い, その結果も成績に反映する。

応用プログラムコースにすすんだ者は 100 点を最高点とし, 基本プログラムコースにとどまった者は 80 点を最高点とし, いずれも最終的に習得した能力で本科目の最終評価点とする。応用プログラムコースは末端再帰や項の処理を理解していることを前提としてすすめる。途中で変更はできないのでよく考えて選択すること。

[基本プログラムコース]

## 演習問題 (r10)[基本プログラムコース]

- (1)  $X, Y$  を  $n$  個の実数から成る長さの等しいリスト  $[X_1, \dots, X_n], [Y_1, \dots, Y_n]$  とする。各  $i$  ( $i = 1, \dots, n$ ) について  $X_i, Y_i$  がともに正の数のときのみ  $X_i, Y_i$  の差をとり, それらの 2 乗の和が  $D$  であるという関係を表す述語  $\text{dist}(X, Y, D)$  のプログラムを作成せよ。
- (2)  $L1, L2$  を要素がすべて整数であるようなリストとする。各  $i$  ( $i = 1, \dots, n$ ) について  $L2$  の  $i$  番目の要素が  $L1$  の  $i$  番目の要素を 3 で割った余りになっているという関係を表す述語  $\text{remlist}(L1, L2)$  のプログラムを作成せよ。たとえば  $\text{remlist}([4, 2, 6], L2)$  は  $L2 = [1, 2, 0]$  となって成功する。
- (3) 空でないリスト  $L$  の最後の要素が '0' であるかどうかを判定する述語  $\text{end\_with\_zero}(L)$  のプログラムを作成せよ。たとえば  $\text{end\_with\_zero}([1, 0])$  は成功し,  $\text{end\_with\_zero}([0, 1])$  は失敗する。
- (4) 2 つのリスト  $L1, L2$  の各要素を組にしたものを要素とするリストが  $L$  であるという関係を表す述語  $\text{make\_pair}(L1, L2, L)$  のプログラムを作成せよ。ただし,  $L1$  と  $L2$  の長さは必ずしも同じではなく, ペアは 2 つの要素がそろうところのみ存在するものとする。たとえば  $\text{make\_pair}([1, 2], [a, b, c], L)$  は  $L = [(1, a), (2, b)]$  となって成功する。
- (5) 以下の文法が定義されているとき, 与えられた項  $\text{Term}$  に  $a, b$  両方ともが出現するかを判定する述語  $\text{both\_occur}(\text{Term})$  のプログラムを作成せよ。ただし,  $\text{Term}$  には文法にかなったものしか入力されないと仮定してよい。また, 必要に応じて自分で新しい述語を定義せよ。たとえば,  $\text{both\_occur}(f(g(a, f(b))))$ ,  $\text{both\_occur}(f(g(0, g(a, b))))$  はいずれも成功する。(対象言語とプログラミング言語を明確に区別して考えること。)

$\text{Term} ::= \text{Alphabet} \mid \text{Digit} \mid f(\text{Term}) \mid g(\text{Term}, \text{Term})$

$\text{Alphabet} ::= a \mid b$

$\text{Digit} ::= 0 \mid 1$

## データの段階的な集積方法

末端再帰型と繰り返し型

例：1 から 5 までの整数のリストを作成．

- 末端再帰型 – リストは前から値がはいっていき，最後に空リストがはい

```
accumulate1(L) :- accum1(1,L).  
  
accum1(6, []).  
accum1(N, [N|X]) :- N1 is N+1, accum1(N1,X).  
  
?- accumulate1(L).  
L = [1,2,3,4,5]
```

- 繰り返し型 – リストは第 3 引数にスタックされていき，最後に第 2 引数に渡される

```
accumulate2(L) :- accum2(1,L, []).  
  
accum2(6,S,S).  
accum2(N,L,S) :- N1 is N+1, accum2(N1,L, [N|S]).  
  
?- accumulate2(L).  
L = [5,4,3,2,1]
```

## 練習問題 [応用プログラムコースのみ]

1. ベクトル  $X = (X_1, \dots, X_n)$ ,  $Y = (Y_1, \dots, Y_n)$  がそれぞれリスト  $[X_1, \dots, X_n]$ ,  $[Y_1, \dots, Y_n]$  で表されるとき， $X, Y$  の内積，すなわち  $\sum_{i=1}^n X_i \cdot Y_i$  が IP であるという関係を表す述語 `inner_product( X, Y, IP )` の再帰型プログラムは以下の通りである．

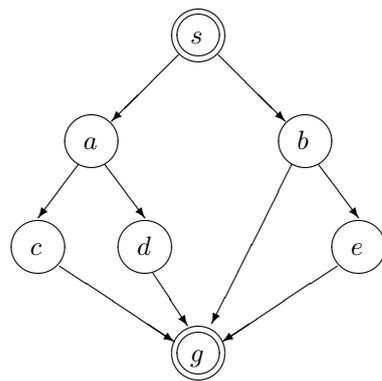
```
inner_product([], [], 0).  
inner_product([X|X1], [Y|Y1], Z) :- inner_product(X1, Y1, Z1), Z is X*Y+Z1.
```

このプログラムの繰り返し型版 `inner_product2( X, Y, IP )` を作成せよ．たとえば，`inner_product2( [1,2], [3,4], IP )` を実行すると，`IP = 11` となって成功する．

## 演習問題 (r10) [応用プログラムコース]

以下の問題においては，必要に応じて新たな述語を定義すること．

- (6) `inner_product` を参考に，リスト  $L$  の長さ  $N$  を求める `list_length2( L, N )` の繰り返し型プログラムを作成せよ．たとえば，`list_length2( [1,2,3], N )` を実行すると， $N = 3$  となって成功する．(Hint: `list_length2` は，スタックに相当する引数が追加された別の述語を呼び出す．)
- (7) リスト  $L1$  の要素がすべて整数であるとする． $L1$  の要素の中でその値が  $Min$  以上  $Max$  以下のものすべてから構成したリストが  $L2$  であるような関係を表す述語 `mid2( L1, Min, Max, L2 )` のプログラムを繰り返し型で作成せよ．たとえば，`mid2( [5,12,2,25,18], 10, 20, L2 )` は  $L2 = [18,12]$  となって成功する．`mid2` は演習問題 r4(2) で作成した `mid` の繰り返し型プログラムであり，要素がリストの後から順に求められるため  $L2$  は `mid` の場合と逆順になる．
- (8)  $L$  をアトムから成る長さ 1 以上のリストだとする． $L$  の要素がすべて同一かどうかを判定する述語 `same_elements(L)` のプログラムを作成せよ．  
たとえば，`same_elements( [a,a,a] )` は成功し，`same_elements( [a,b,a] )` は失敗する．  
この問題については普通の末端再帰型でプログラムしてください．
- (9) ループのない有向グラフとそのノード 2 つが初期ノード，目標ノードとして与えられたとき，初期ノードから目標ノードに到る経路  $Path$  を縦型探索 (深さ優先探索) を使って求める `dfs2(Path)` の繰り返し型プログラムを作成せよ．`dfs2` は演習問題 r9(5) で作成した `dfs` の繰り返し型プログラムであり，要素がボトムアップに求められるため  $Path$  は `dfs` の場合と逆順になる．(r9 練習問題 2, r9(5) と同様に，`start(S)`, `goal(G)`, `edge(N,M)` を使用してグラフの形状を記述して考えよ．)  
さらに，得られたすべての径路の集合が  $PathList$  であるという関係を表す述語 `dfs2_list(PathList)` のプログラムを `setof` を使って作成せよ．図 10.1 のグラフで全解が求まることを確認せよ．`dfs2_list( L )` は  $L = [ [g,b,s], [g,c,a,s], [g,d,a,s], [g,e,b,s] ]$  となって成功する．
- (10) (9) のプログラムを改訂して初期ノードから目標ノードに到る経路  $Path$  の探索の深さを  $MaxDepth$  までに限定した縦型探索を行う `dfs_bound( MaxDepth, Path )` の繰り返し型プログラムを作成せよ．さらに，得られたすべての径路の集合が  $PathList$  であるという関係を表す述語 `dfs_bound_list( MaxDepth, PathList )` のプログラムを `setof` を使って作成せよ．図 10.1 のグラフで深さ 1,2,3,4 それぞれについて  $Path$  が求まることを確認せよ．たとえば，`dfs_bound_list(2,L)` は  $L = [ [g, b, s] ]$  となって成功する．
- (11)\* 第 13 回実習では応用プログラムを作成する．とりあげてほしい題材があれば考慮するので記述してください (ただし実現できるかどうかはわからない)



▣ 10.1