プログラム言語論　平成22年度前学期試験
2010年7月　沼尾雅之

プログラム言語論　試験問題

## 問題1

命令型言語，オブジェクト指向型言語，関数型言語の特徴を比較した下記の表を完成させよ．ただし，説明には，下記のキーワードリスト及びプログラミング言語リストから単語を適宜選択し，使用した単語には下線を引くこと．また，複数個所に同じ単語を使用するのはかまわないが，下線を引くのは，リスト中の単語1つにつき，最も重要と思う1個所のみとする．

**キーワードリスト** 手続き型言語，コンパイラ，インタプリタ，Read-Eval-Print，数値計算，記号処理，人工知能，シミュレーション，自然言語処理，システム（OS）記述，統合プログラミング環境，多様性，クラス，メンバ変数，局所変数，抽象クラス，再帰関数，継承，高階関数，ラムダ計算，手続きと関数，代入文，フォンノイマン型計算機，宣言文，実行文，オーバーロード，オーバーライド，制御文，S式，リスト記法，項，述語，前置記法，カプセル化，情報隠蔽，ランダムアクセスマシン，構造化プログラミング，関係，規則と事実，質問，単一化，バックトラッキング，一階述語論理

**プログラミング言語リスト** C，C++，Pascal，Java，Scheme，Lisp，Prolog，Smalltalk，Guile，GNU-Prolog，Fortran

| | 計算モデル | 文法 | 用途 | 言語処理系 |
|---|---|---|---|---|
| 命令型言語 | (P1) | (P2) | (P3) | (P4) |
| オブジェクト指向型言語 | (O1) | (O2) | (O3) | (O4) |
| 関数型言語 | (F1) | (F2) | (F3) | (F4) |
| 論理型言語 | プログラムは，パラメータ間の 関係 を規定するための 規則と事実 から構成され，質問 によって実行が起動される．一階述語論理 の計算モデルに基づき，質問に対する解を 単一化 や バックトラッキング によって自動的に探索する | アトム，変数，数を表す 項 と，関数，述語 を表す複合項から構成される．大文字で始まる文字列は変数として扱われる． | 人工知能，自然言語処理 等の記号処理に用いられる | Prolog，GNU-Prolog |

## 問題2

Lisp と Prolog における文字列処理プログラムを比較する．たとえば，リスト $L$ の先頭要素を求める Lisp 関数 $car(L)$ に対する Prolog 述語 $car(L, CAR)$ は以下のように定義される．（大文字変数やアンダーバーやピリオドに注意．）

$$car([CAR|\_], CAR).$$

このように Prolog では，Lisp における関数の入力だけでなく出力も1つのパラメータにして，入出力の関係を示す述語として定義される．

(1) Lisp でのリスト表現 $(a\ b\ (c\ d))$ を，ドット表現に直せ．
(2) Lisp 関数 $cdr(L), cons(A, L)$ に対応する Prolog 述語を定義せよ．
(3) 2つの文字列用変数 s1,s2 を連結するための Lisp 関数 $(append\ s1\ s2)$ を定義せよ．

(4) 3つの文字列用変数 S1,S2,S3 について，文字列 S3 が文字列 S1 と S2 の連結になっている関係を満たす Prolog 述語 $append(S1, S2, S3)$ を定義せよ．

(5)Prolog の質問？ $-\ append(S1, S2, [p,\ q])$. に対する解を全て求めよ．

## 問題 3

以下の C++プログラムについて下記の問いに答えよ．

```
#include <iostream>
using namespace std;

class Atom {
public:
  virtual void print() {};          /*   (A)   */
};

class List {
  Atom* element;
  List* next ;
public:
  List(Atom* a, List* n = NULL)
    { element = a; next = n; }
  Atom* car() { return element; }
  List* cdr() { return next; }
  void printdot();
  void printlist(List* l);
};

List* cons(Atom* a, List* l)
{ List* n = new List(a,l); return n; };

class Int: public Atom {
  int value;
public:
  Int(int i) { value = i; }
  void print() { cout << value; }
};

class Symbol: public Atom {
  char* value;
public:
  Symbol(char* s) {
    value = new char[strlen(s)+1];  /*   (B)   */
    strcpy(value,s); }
  void print() { cout << value; }
};
```

```
void printlist(List* l) {
  List* ln = l;
  cout << "(";
  while(ln != NULL) {
    ln->car()->print();
    cout << " ";
    ln = ln->cdr();
  }
  cout << ")";
};

void List::printdot() {
        /*   (C)   */
};

List* append(List* l1, List* l2) {
        /*   (D)   */
};

List* reverse(List* l1, List* l2) {
        /*   (E)   */
};

main()
{
  Int* i2010 = new Int(2010);
  Symbol* sUEC = new Symbol("UEC");
  Symbol* sto = new Symbol("to");
  Symbol* sWelcome = new Symbol("Welcome");
  List* list1 = new List(i2010);
  list1 = cons(sWelcome, cons(sto, cons(sUEC, list1)));
  printlist(list1);                    /*   (F)   */
  list1->printdot();
  printlist(append(list1,list1));
  printlist(reverse(list1,NULL));
}
```

(1) コメント (A) の *virtual* の意味を，それが無かった場合のプログラムの挙動とともに説明せよ．

(2) コメント (B) で $value = s$; としてはいけない理由を説明せよ．

(3) コメント (F) の関数 $printlist(list1)$ の出力結果を示せ．

(4) コメント (C) のメンバ関数 $printdot()$ の定義を，なるべく再帰関数によって完成させよ．これは，リスト構造をドット表現で出力するものである．

(5) コメント (D) の関数 $append(List*l1, List*l2)$ の定義を，なるべく $cons, car, cdr$ と再帰を用いて完成させよ．これは，引数で示される2つのリスト構造を接続するものである．たとえば，$(a\ b)$ と $(c\ d)$ を接続すると $(a\ b\ c\ d)$ となる．

(6) コメント (E) の関数 $reverse(List*l1, List*l2)$ の定義を，なるべく $cons, car, cdr$ と再帰を用いて完成させよ．これは，第一引数で示されるリスト構造を反転させ，第二引数に接続するものである．たとえば，$(a\ b\ c)$ を反転させると $(c\ b\ a)$ となる．通常，第二引数には NULL を入れて実行する．

(7) 2つの関数 *append* と *reverse* の実行の違いを再帰法の観点から説明せよ．

以上