プログラム言語論・テスト問題用紙

('06 年 2 月 14 日（火）・ 8:50 〜 10:20)

解答上、その他の注意事項

I. 問題は、問 I〜V まである。

II. 解答用紙の右上の欄に学籍番号・名前を記入すること。

III. 解答欄を間違えないよう注意すること。

IV. 解答中の文字 (特に a と d) がはっきりと区別できるよう注意すること。

V. 持ち込みは不可である。筆記用具・時計・学生証以外のものは、かばんの中などにしまうこと。

VI. テストの配点は 80 点である。合格はレポートの得点を加点して、100 点満点中 60 点以上とする。

I. （Backus-Naur 記法）

下の記号列の中で、次の BNF

$S \rightarrow S\ S\ +\ |\ S\ S\ *\ |\ \mathtt{x}$

の 非終端記号 S から生成されるものには○、生成されないものには×をつけよ。

(1) `x x + x x *`　　(2) `x x x + *`

(3) `x + x * x`　　(4) `x x * x +`

II. （コンパイラのフェーズ）

次の (1)〜(3) の C 言語のプログラムにはそれぞれ誤りがある。コンパイラのどのフェーズで誤りが検出されるか？（あるいはされないか？） もっとも適当なものを下の選択肢 (A)〜(E) から選べ。

(1) （ポインタ型変数に浮動小数点数を代入しようとした。）

```
#include <stdio.h>
int main(void) {
    double *p;
    p = 2.1;
    printf("%f¥n", p);
    return 0;
}
```

(2) （文字列リテラルの「"」を「'」と書き間違えた。）

```
#include <stdio.h>
int main(void) {
    printf('Hello! World¥n');
    return 0;
}
```

(3) （for 文の「;」を「,」と書き間違えた。）

```
#include <stdio.h>
int main(void) {
    int x;
    for(x=10, x>0, x--) {
        printf("Hello! World¥n");
    }
    return 0;
}
```

(1)〜(3) の選択肢

(A) 字句解析フェーズでエラーが検出される。

(B) 構文解析フェーズでエラーが検出される。

(C) 意味解析フェーズでエラーが検出される。

(D) コード生成フェーズでエラーが検出される。

(E) 実行時にエラーとなるか、まったくエラーにならない（が作成者の意図と異なる動作をする）。

III. （正規表現）

「a(a|bab)*」という正規表現に（一部でなく）全体がマッチする文字列には (A)、「a(ba)*b」という正規表現に全体がマッチする文字列には (B)、両方に全体がマッチする文字列には (C)、どちらにもマッチしない文字列には (D) をつけよ。

(1) ababaab　(2) abababab　(3) ababbab　(4) ababaabab

IV. （演算子順位法）

次の BNF で表される文法は、（C 言語の）論理式を表している。

$E \rightarrow$ **id** | “(” E “)” | E “&&” E | E “||” E | “!” E

ただし、この文法は曖昧なので、優先順位と結合性について次のように決めておく。

> 「&&」と「||」は、いずれも左結合で、「!」は「&&」 よりも優先順位が高く、「&&」は「||」よりも優先順位が高いものとする。つまり、!a&&b は (!a)&&b と解釈され、a || b && c は a || (b && c) と解釈される。

以下の演算子順位行列の空欄を ⋖（シフト）, ⋗（還元）, ≒（特別なシフト）, **err**（エラー）のいずれかで埋めよ。

左 \ 右	||	&&	!	(	)	**id**	終
始	⋖	⋖	⋖	⋖	**err**	⋖	≒
||	⋗	(1)	⋖	⋖	⋗	⋖	⋗
&&	(2)	(3)	⋖	⋖	⋗	⋖	⋗
!	⋗	⋗	⋖	⋖	⋗	⋖	⋗
(	⋖	⋖	⋖	⋖	≒	⋖	**err**
)	⋗	⋗	**err**	(4)	⋗	**err**	⋗
id	⋗	⋗	**err**	**err**	⋗	**err**	⋗

V. （再帰下降構文解析）

次のような BNF で表された文法に対して、再帰下降構文解析ルーチンを作成する。

$$
\begin{array}{lcl}
S & \rightarrow & \text{“(”}\ L\ \text{“)”} \\
 & | & \mathbf{a} \\
L & \rightarrow & S\ L' \\
L' & \rightarrow & \text{“,”}\ S\ L' \\
 & | & \boldsymbol{\varepsilon}
\end{array}
$$

ただし、「S」,「L」,「L'」は非終端記号、「**a**」,「(」,「)」と「,」は終端記号である。また、開始記号は「S」である。

(1) この生成規則のうち、

$$
\begin{array}{lcl}
L & \rightarrow & S\ L' \\
L' & \rightarrow & \text{“,”}\ S\ L' \\
 & | & \boldsymbol{\varepsilon}
\end{array}
$$

の部分は、次の生成規則

$$
\begin{array}{lcl}
L & \rightarrow & L\ \text{“,”}\ S \\
 & | & S
\end{array}
$$

から、左再帰を除去したものである。これにならって次の生成規則から左再帰を除去せよ。

$$
\begin{array}{lcl}
T & \rightarrow & T\ \text{“+”}\ F \\
 & | & F
\end{array}
$$

ただし、補助的な非終端記号としては T' を用いること。

(2) *Follow*(L') — L' の直後に現れうる終端記号の集合 — を求めよ。

(3) 〜 (5)

この文法に対して、入力が文法にしたがっていれば「正しい構文です。」間違っていれば「構文に誤りがあります。」と表示する構文解析プログラムを作成する。プログラム（次ページ）中の空欄を埋めよ。ただし、非終端記号 S, L, L' に対応する関数は、それぞれ S, L, L1 である。

（**プログラムの補足説明:** プログラム中では、終端記号は、“(”, “)”, “,” のような記号は、その字そのもの（の ASCII コード）、**a** は、C 言語の マクロ `ALPHA` として表現している。`yylex` 関数は、入力を読んで、次の終端記号を返す関数である。`token` という大域変数に、現在処理中の終端記号が代入される。`eat` 関数は、現在 `token` に入っている値が、引数として与えられた終端記号と等しいかどうか確かめ、等しければ次の終端記号を読み込む。）

再帰下降構文解析プログラム

```
... /* これより上の部分は後掲 */

/* 関数プロトタイプ宣言 */
void S(void);
void L(void);
void L1(void);

/* 再帰的構文解析関数群
   文法の各非終端記号に対応する関数 */

void S() {
    if (token == '(') {
        (3)
    } else if (token == ALPHA) {
        eat(ALPHA);
    } else {
        printf("構文に誤りがあります。¥n");
        exit (0); /* プログラムを終了 */
    }
}

void L() {
    S(); L1();
}

void L1() {
    if ( (4) ) {
        eat(','); S(); L1();
    } else if ( (5) ) {
        /* 何もしない */
    } else {
        printf("構文に誤りがあります。¥n");
        exit (0); /* プログラムを終了 */
    }
}

/* main関数 */
int main() {
    token = yylex(); /* 最初のトークンを読む */
    S();
    printf("正しい構文です!¥n");
}
```

（参考） 上のプログラムで省略された最初の部分（問題には直接は関係ない。）

```
#include <stdio.h>
#include <stdlib.h>    /* exit() 用 */
#include <ctype.h>     /* isalpha() 用 */

#define ALPHA   256
/* 大域変数の宣言 */
int token;

int yylex() {   /* 簡易字句解析ルーチン */
    int c;
    char buf[256];

    do { /* 空白は読みとばす。*/
        c = getchar();
    } while (c == ' ' || c == '¥t' || c == '¥n');

    if (isalpha(c)) { /* アルファベットだったら ALPHA を返す。*/
        return ALPHA;
    } else if (c == EOF) {
        return EOF;
    } else {
    /* 上のどの条件にも合わなければ、文字をそのまま返す。*/
        return c;   /* '(', ')', ',' など */
    }
}

void eat(int t) {     /* token（終端記号）を消費して、次の token を読む */
    if (token == t) {
        token = yylex();
        return;
    } else {
        printf("構文に誤りがあります。¥n");
        exit (0);
    }
}
```

プログラム言語論・テスト解答用紙 ('06 年 2 月 14 日)

学籍番号		氏名	

I. （Backus-Naur 記法） (4×4)

(1).		(2).		(3).		(4).	

II. （コンパイラのフェーズ） (4×3)

(1).		(2).		(3).	

III. （正規表現） (4×4)

(1).		(2).		(3).		(4).	

IV. （演算子順位法） (4×4)

(1).		(2).		(3).		(4).	

V. （再帰下降構文解析） (4×5)

(1).

$T \rightarrow$

(2).	{ }
(3).	
(4).	
(5).	

テストの感想