# プログラム言語論（’09年度）・期末テスト問題用紙

（’10年2月17日（水）・ 18:00 〜 19:30）

## 解答上、その他の注意事項

I. 問題は、問I〜VI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）

次のような BNF で表される文法を考える。

$$
\begin{array}{rcl}
S & \rightarrow & \texttt{+}\,S\;S \\
 & | & \texttt{*}\,S\;S \\
 & | & \texttt{x}
\end{array}
$$

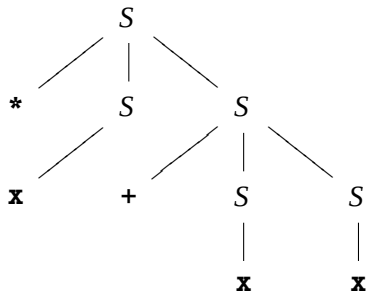
次の各記号列について、上の BNF の非終端記号 *S* から導出されるものには、その解析木 (parse tree) を右の例にならって書き、導出されないものには ✗ を記せ。（解析木は一通りとは限らないが、そのうちひとつを書けば良い。）

(1) `* * x x x`

(2) `* x x + x`

(3) `+ + x x * x x`

**例:** `* x + x x` に対する解析木




II. （正規表現）

「`(ab|ba)*(a|`$\varepsilon$`)`」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 (L) を、
「`(aba|bab)*(b|`$\varepsilon$`)`」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 (R) を、
両方に全体がマッチする文字列には (B) を、
どちらにも全体がマッチしない文字列には (N) を記せ。

(1) `ababaabab`

(2) `abababababa`

(3) `abababbbab`

III. （コンパイラのフェーズ）

コンパイラは、字句（単語）を切り分ける字句解析フェーズ、プログラムの構造を木の形に表す構文解析フェーズ、変数の宣言や型のチェックを行なう意味解析（静的解析）フェーズ、目的のコードを生成するコード生成フェーズなどに概念的に分けることができる。

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最初に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。なお、(1)〜(3) のいずれも単独でコンパイルされ、標準ライブラリとのみリンクされるものとする。（つまり、他のファイルに変数や関数が定義されていることはない。）

(1) （文字列リテラルに二重引用符「"」をつけるのを忘れた。）

```
#include <stdio.h>

int main(void) {
    printf(Hello);
    return 0;
}
```

(2) （文末のセミコロン「;」を忘れた。）

```
#include <stdio.h>

int main(void)
    printf("Hello World!¥n")
    return 0;
}
```

(3) （コメントを閉じるのを忘れた。）

```
#include <stdio.h>

int main(void) { /* コメント
    printf("Hello");
    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）。

IV. （演算子順位法）

次の BNF で表される文法を演算子順位法により構文解析する。

$$E \rightarrow \quad \textbf{id} \;\mid\; E \text{ “..” } E \;\mid\; E \text{ “=” } E \;\mid\; E \text{ “,” } E \;\mid\; \text{“(” } E \text{ “)”}$$

ただし、**id** はアルファベット 1 文字からなるトークンを表す。

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。

「..」は非結合、「=」は右結合、「,」は左結合であり、「..」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、下表中の左の欄の式は、右の欄の式として解釈される。

| 式 | 解釈 |
|---|---|
| `a .. b .. c` | 構文エラー |
| `a = b = c` | `a = (b = c)` |
| `a , b , c` | `(a , b) , c` |
| `a .. b = c` | `(a .. b) = c` |
| `a = b .. c` | `a = (b .. c)` |
| `a .. b , c` | `(a .. b) , c` |
| `a , b .. c` | `a , (b .. c)` |
| `a = b , c` | `(a = b) , c` |
| `a , b = c` | `a , (b = c)` |

以下の演算子順位行列の空欄 (1)〜(4) を ⋖（シフト）、 ⋗（還元）、✗（エラー）のうちもっとも適切なもので埋めよ。

| 左 \ 右 | .. | = | , | ( | ) | **id** | 終 |
|---|---|---|---|---|---|---|---|
| 始 | ⋖ | ⋖ | ⋖ | ⋖ | ✗ | ⋖ | ≐ |
| .. | (1) | ⋗ | ⋗ | ⋖ | ⋗ | ⋖ | ⋗ |
| = | (2) | (3) | ⋗ | ⋖ | ⋗ | ⋖ | ⋗ |
| , | ⋖ | ⋖ | (4) | ⋖ | ⋗ | ⋖ | ⋗ |
| ( | ⋖ | ⋖ | ⋖ | ⋖ | ≐ | ⋖ | ✗ |
| ) | ⋗ | ⋗ | ⋗ | ✗ | ⋗ | ✗ | ⋗ |
| **id** | ⋗ | ⋗ | ⋗ | ✗ | ⋗ | ✗ | ⋗ |

V. （再帰下降構文解析）

次のような BNF で定義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。

$$
\begin{array}{lcl}
L & \rightarrow & A\ L' \\
L' & \rightarrow & \text{“||”}\ A\ L' \quad | \quad \varepsilon \\
A & \rightarrow & N\ A' \\
A' & \rightarrow & \text{“\&\&”}\ N\ A' \quad | \quad \varepsilon \\
N & \rightarrow & \text{“!”}\ N \quad | \quad \text{“(”}\ L\ \text{“)”} \quad | \quad \mathtt{x}
\end{array}
$$

ただし、「$L$」,「$L'$」,「$A$」,「$A'$」,「$N$」は非終端記号で、「||」,「**&&**」,「**!**」,「**(**」,「**)**」,「**x**」は終端記号とする。開始記号（start symbol）は $L$ である。

(1) *First*($A$) を求めよ。

(2) *Follow*($A'$) を求めよ。

(3) 下の構文解析表の $L$ の行を埋めよ。

(4) 下の構文解析表の $A'$ の行を埋めよ。

| | || | && | ! | ( | ) | x | $ |
|---|---|---|---|---|---|---|---|
| $L \rightarrow$ | ? | ? | ? | ? | ? | ? | ? |
| $L' \rightarrow$ | “||” $A\ L'$ | ✗ | ✗ | ✗ | $\varepsilon$ | ✗ | $\varepsilon$ |
| $A \rightarrow$ | ✗ | ✗ | $N\ A'$ | $N\ A'$ | ✗ | $N\ A'$ | ✗ |
| $A' \rightarrow$ | ? | ? | ? | ? | ? | ? | ? |
| $N \rightarrow$ | ✗ | ✗ | “!” $N$ | “(” $L$ “)” | ✗ | x | ✗ |

ただし、✗ の欄は “構文誤り” を示す。

(3), (4) の解答は次の選択肢から選べ。

(A). $\varepsilon$　　(B). “&&” $N\ A'$　　(C). ✗　　(D). $A\ L'$

VI. （LR 構文解析）

「^」,「_」などの演算子はあるテキスト整形言語で使われている演算子で、`x^a` は上付きの添字 $x^a$、また `x_a` は下付きの添字 $x_a$ を表す。このテキスト整形言語では `x_a^b` を特別扱いして、これを $x_{a^b}$ や ${x_a}^b$ ではなく、$x_a^b$ のように整形する。

このことを踏まえて...

次のような文法（··· の後は生成規則の番号）

$$
\begin{array}{rcll}
E & \to & E \text{ “\_” } E \text{ “\^{}” } E & \cdots I \\
 & | & E \text{ “\_” } E & \cdots II \\
 & | & E \text{ “\^{}” } E & \cdots III \\
 & | & \text{“\{” } E \text{ “\}”} & \cdots IV \\
 & | & \mathbf{a} & \cdots V
\end{array}
$$

に対して、LR 構文解析表を作成する。ただし、

- ··· の後の I, II などは生成規則の番号である。
- 「$E$」は非終端記号である。
- 「_」,「^」,「{」,「}」,「**a**」は終端記号である。このうち、「**a**」はアルファベット 1 文字からなるトークンを表す。
- 「^」,「_」演算子の優先度は等しく、どちらも右結合である。

`bison` の出力する LR 構文解析表は次のようになる。（**注:** `bison` に `-v` オプションを指定することによって、LR 構文解析表をファイルに出力させることができる。）

| | _ | ^ | { | } | **a** | $ | *E* |
|---|---|---|---|---|---|---|---|
| ⓪ | | | shift ② | | shift ① | | goto ③ |
| ① | reduce V | | | | | | |
| ② | | | shift ② | | shift ① | | goto ④ |
| ③ | shift ⑥ | shift ⑤ | | | | accept | |
| ④ | shift ⑥ | shift ⑤ | | shift ⑦ | | | |
| ⑤ | | | shift ② | | shift ① | | goto ⑧ |
| ⑥ | | | shift ② | | shift ① | | goto ⑨ |
| ⑦ | reduce IV | | | | | | |
| ⑧ | shift ⑥ | shift ⑤ | reduce III | | | | |
| ⑨ | shift ⑥ | shift ⑩ | reduce II | | | | |
| ⑩ | | | shift ② | | shift ① | | goto ⑪ |
| ⑪ | shift ⑥ | shift ⑤ | ? ? ? ? ? ? | | | | |

**注:**
shift ⓢは、「シフトして状態ⓢへ遷移」、
goto ⓢは、「状態ⓢへ遷移」、
reduce X は、「生成規則 X を使って還元」を表す。

(1) 〜 (2)

次の入力に対して、↑の次（右）の記号を**シフトした直後**の（つまりシフトしたあと、還元がまだ起こっていない時の）スタックの状態はどのようになっているか?

(1) `{aˆbˆc}`　　(2) `{{aˆb}ˆc}`

下の選択肢（(1)〜(2) 共通）から選べ。（左がスタックの底とする）

(A). ⓪{②E④ˆ⑤　　(B). ⓪{②E④ˆ⑤E⑧ˆ⑤

(C). ⓪{②{②E④ˆ⑤**a**①}⑦ˆ⑤　　(D). ⓪{②{②E④ˆ⑤E⑧}⑦ˆ⑤

(3) `a_bˆc` という入力に対しては、`c` をシフトしたあと 1 回生成規則 V を用いて還元を行なって、⓪E③_⑥E⑨ˆ⑩E⑪ というスタックの状態になる。「還元還元衝突（reduce/reduce conflict）の時は、上（先）に書かれている構文規則が優先する。」という yacc/bison の 衝突回避規則に従うと、表の？？？？？？の部分には何が入るべきか、次の選択肢から選べ。

(A). shift ④　　(B). shift ⑦　　(C). reduce I　　(D). reduce III

# プログラム言語論（’09 年度）・期末テスト解答用紙 (’10 年 2 月 17 日)

| 学籍番号 | | 氏名 | |
|---|---|---|---|

I. （Backus-Naur 記法） (4×3)

| (1). | (2). | (3). |
|---|---|---|
| | | |

II. （正規表現） (4×3)

| (1). | | (2). | | (3). | |
|---|---|---|---|---|---|

III. （コンパイラのフェーズ） (4×3)

| (1). | | (2). | | (3). | |
|---|---|---|---|---|---|

IV. （演算子順位法） (4×4)

| (1). | | (2). | | (3). | | (4). | |
|---|---|---|---|---|---|---|---|

裏ページに続く。

V. （再帰下降構文解析） (4×4)

| | |
|---|---|
| (1). | { |
| (2). | { |

| | || | && | ! | ( | ) | x | $ |
|---|---|---|---|---|---|---|---|
| (3). $L \rightarrow$ | | | | | | | |
| (4). $A' \rightarrow$ | | | | | | | |

VI. （LR 構文解析） (4×3)

| | | | | | |
|---|---|---|---|---|---|
| (1). | | (2). | | (3). | |

テストの感想