コンパイラ・期末テスト問題用紙
（2021年 02月 09日・ 10:30 ～ 12:00）

解答上、その他の注意事項

1. 問題は、問 Ⅰ～Ⅳ まである。
2. 解答用紙の右上の欄に学籍番号・名前を記入すること。
3. 解答欄を間違えないよう注意すること。
4. 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。
5. 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。
6. テストの配点は 80 点である。合格は毎週の課題の得点を加点して、100 点満点中 60 点以上とする。

I. （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。

$$E \to \quad \mathbf{id} \;\mid\; E \text{ “>” } E \;\mid\; E \text{ “&&” } E \;\mid\; E \text{ “..” } E \;\mid\; E \text{ “=” } E \;\mid\; \text{“(” } E \text{ “)”}$$

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、「&&」は左結合、「..」は非結合、「=」は右結合、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	（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)

式	解釈
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) ～ (10) を ⋖、≐、⋗、✗のうちもっとも適切なもので埋めよ。ただし ✗はエラーを表すものとする。（教科書などの記法では、エラーは空欄のままとしているが、このテストでは無回答と区別するために明示的に✗を書くことにする。）

左＼右	=	..	&&	>	(	)	**id**	終
始	⋖	⋖	⋖	⋖	⋖	✗	⋖	≐
=	(1)	⋖	⋖	(2)	⋖	(3)	⋖	⋗
..	(4)	(5)	⋖	⋖	⋖	⋗	⋖	⋗
&&	⋗	⋗	(6)	(7)	⋖	⋗	⋖	⋗
>	⋗	⋗	(8)	(9)	⋖	⋗	⋖	⋗
(	⋖	⋖	⋖	(10)	⋖	≐	⋖	✗
)	⋗	⋗	⋗	⋗	✗	⋗	✗	⋗
id	⋗	⋗	⋗	⋗	✗	⋗	✗	⋗

II. （再帰下降構文解析）

対のような 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。

$$
\begin{array}{lcll}
S & \to & \mathbf{a}\ \text{“=”}\ E\ \text{“;”} & \cdots ① \\
 & | & \mathbf{if}\ \text{“(”}\ E\ \text{“)”}\ S\ \mathbf{else}\ S & \cdots ② \\
 & | & \mathbf{while}\ \text{“(”}\ E\ \text{“)”}\ S & \cdots ③ \\
 & | & \text{“\{”}\ T\ \text{“\}”} & \cdots ④ \\
T & \to & S\ T & \cdots ⑤ \\
 & | & \varepsilon & \cdots ⑥ \\
E & \to & \mathbf{a} \mid \text{“(”}\ E\ \text{“)”} \mid E\ \text{“(”}\ E\ \text{“)”} \mid E\ \text{“[”}\ E\ \text{“]”} &
\end{array}
$$

ただし、「S」，「T」，「E」は非終端記号、「$\mathbf{a}$」，「=」，「;」，「$\mathbf{if}$」，「(」，「)」，「$\mathbf{else}$」，「$\mathbf{while}$」，「{」，「}」，「[」，「]」 は終端記号である。 開始記号（start symbol）は S である。

(1) S から導出される終端記号の列で、次の条件を満たすものの例を挙げよ。存在しなければ✗を記せ。

(i) 「;」の直後に「}」が続く。
(ii) 「;」で終わる。
(iii) 「]」の直後に「}」が続く。
(iv) 「]」で終わる。

(2) E から左再帰を除去せよ。補助的に導入する非終端記号は E' とせよ。（後の解答で使用するために、生成規則に丸数字（⑦, ⑧, …）を付けておくこと。）

以下の問は (2). で E から左再帰を除去して得られた BNF について答えよ。

(3) $First(S)$ を求めよ。

(4) $Follow(T)$ を求めよ。

(5) $Follow(E')$ を求めよ。

(6) 下の予測型構文解析表の S, T の行を埋めよ。この問題の解答は✗, ① ～ ⑥ の中から選べ。ただし、✗は“エラー”を示す。（教科書などの記法では、エラーは空欄のままとしているが、このテストでは無回答と区別するために明示的に✗を書くことにする。）

(7) 下の予測型構文解析表の E, E' の行を埋めよ。この問題の解答は✗と ⑦, ⑧, …（(2) の解答で、BNF の生成規則に自分で付けた番号）から選べ。構文エラーの場合は、必ず✗を記入し、空欄のまま残さないこと。

	a	=	;	**if**	(	)	**else**	**while**	{	}	[	]	**$**
$S \to$													
$T \to$													
$E \to$													
$E' \to$													

(8) 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）中の指定の部分に入る S, T, E, E1 関数のうち、E, E1関数の定義を完成させよ。ただし、S, T, E, E1 は、それぞれ非終端記号 S, T, E, E' に対応する関数である。予測型構文解析表の✗に相当する入力には `reportError` 関数を呼び出すようにすること。

（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 1 文字のものは、その字そのもの（の ASCII コード）、**if** などのトークンは、C 言語のマクロ（例えば **if** の場合は IF）として表現している。入力の終わり (\$) に対応するのは、このプログラムの場合、マクロ EOF である。

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を代入する。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。reportError 関数は、「構文に誤りがあります。」と表示し、プログラムを終了する。）

再帰下降構文解析プログラム

```c
#include <stdio.h>    /* printf(), EOF など */
#include <stdlib.h>   /* exit()用    */
#include <string.h>   /* strcmp()用  */
#include <ctype.h>    /* isalpha()用 */

/* 終端記号に対するマクロの定義 */
#define A     257    /* トークン a */
#define IF    258    /* トークン if */
#define ELSE  259    /* トークン else */
#define WHILE 260    /* トークン while */

int token;           /* 大域変数の宣言 */

/* 関数プロトタイプ宣言 */
void reportError(void);
int  yylex(void);
void eat(int t);

void S(void);
void T(void);
void E(void);
void E1(void);

/* ************************************************************** */
 * この部分に 関数 S, T, E, E1 の定義を挿入する。                    */
/* ************************************************************** */

/* ここ以降は解答に直接関係はない。 */
void reportError(void) {
  printf("構文に誤りがあります。\n"); exit(0); /* プログラムを終了 */
}

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

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

  do { /* 空白は読み飛ばす。 */
    c = getchar();
  } while (c == ' ' || c == '\t' || c == '\n');

  if (isalpha(c)) {  /* アルファベットだったら… */
    char* ptr = buf;
    ungetc(c, stdin);
    while (1) {
      c = getchar();
```

```
      if (!isalpha(c) && !isdigit(c)) break;
      *ptr++ = c;
    }
    *ptr = '\0';
    ungetc(c, stdin);

    if (strcmp(buf, "a") == 0)     return A;
    if (strcmp(buf, "if") == 0)    return IF;
    if (strcmp(buf, "else") == 0)  return ELSE;
    if (strcmp(buf, "while") == 0) return WHILE;
    reportError();
  } 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) {
    /* 現在のトークンを捨てて、次のトークンを読む */
    token = yylex();
    return;
  } else {
    reportError();
  }
}

```

III. （LR 構文解析）

次のような BNF で与えられる文法

$$
\begin{array}{lcll}
E & \to & \mathbf{id} & \cdots \text{ I} \\
 & | & \text{“\{”} \; X \; \text{“\}”} & \cdots \text{ II} \\
 & | & E \; \text{“\{”} \; X \; \text{“\}”} & \cdots \text{ III} \\
 & | & E \; \text{“\#”} \; \mathbf{id} & \cdots \text{ IV}
\end{array}
\qquad
\begin{array}{lcll}
X & \to & \mathbf{id} \; \text{“=”} \; E & \cdots \text{ V} \\
 & | & X \; \text{“;”} \; \mathbf{id} \; \text{“=”} \; E & \cdots \text{ VI}
\end{array}
$$

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」,「X」は非終端記号である。「$\mathbf{id}$」,「{」,「}」,「#」,「=」,「;」は終端記号である。このうち、「$\mathbf{id}$」はアルファベット 1文字からなるトークンを表す。
- 開始記号（start symbol）は E 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。）

	id	{	}	#	=	;	$		E	X
⓪	s ①	s ②							g ③	
①	r Ⅰ									
②	s ④									g ⑤
③		s ⑦		s ⑧			s ⑥			
④					s ⑨					
⑤			s ⑩			s ⑪				
⑥	accept									
⑦	s ④									g ⑫
⑧	s ⑬									
⑨	s ①	s ②							g ⑭	
⑩	r Ⅱ									
⑪	s ⑮									
⑫			s ⑯			s ⑪				
⑬	r Ⅳ									
⑭	r Ⅴ	s ⑦	r Ⅴ	s ⑧	r Ⅴ					
⑮					s ⑰					
⑯	r Ⅲ									
⑰	s ①	s ②							g ⑱	
⑱	r Ⅵ	s ⑦	r Ⅵ	s ⑧	r Ⅵ					

注: ここで、 s ㉑ は、「シフト (shift) して状態 ㉑ へ遷移」、 g ㉓ は、「状態 ㉓ へ遷移 (go)」、 r Ⅻ は、「生成規則 Ⅻ を使って還元 (reduce)」を表す。

次の入力列に対して、下線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ときの）スタックの状態はどのようになっているか？

(1) a#b#c　　(2) {a=b}{x=y;z=w}　　(3) a{a=b}{x=y#z}　　(4) a{x=y;z=w;u=v}

下の選択肢（(1) ～ (4) 共通）から選べ。（左がスタックの底とする。）

(A). ⓪E③{⑦**id**④　(B). ⓪E③#⑧**id**⑬　(C). ⓪E③{⑦X⑫;⑪**id**⑮
(D). ⓪E③{⑦**id**④=⑨E⑭;⑪**id**⑮　(E). ⓪E③{⑦**id**④=⑨E⑭#⑧**id**⑬
(F). ⓪E③{⑦**id**④=⑨E⑭}⑯{⑦**id**④　(G). ⓪E③{⑦X⑫;⑪**id**④=⑨E⑭;⑪**id**⑮
(H). ⓪E③{⑦**id**④=⑨E⑭;⑪**id**④=⑨E⑭;⑪**id**⑮
(I). ⓪E③{⑦**id**④=⑨E⑭}⑯{⑦**id**④=⑨E⑭}⑯{⑦**id**④

コンパイラ・期末テスト解答用紙（**2021** 年 **02** 月 **09** 日）

学籍番号		氏名	

I. (2 × 10)

(1)		(2)		(3)		(4)		(5)	
(6)		(7)		(8)		(9)		(10)	

II. (2 × 4, 5, 3, 3, 3, 6, 6, 6)

(1-ⅰ)	
(1-ⅱ)	
(1-ⅲ)	
(1-ⅳ)	
(2)	$E \quad \rightarrow$ $E' \quad \rightarrow$
(3)	
(4)	
(5)	

(6)

	a	=	;	**if**	(	)	**else**	**while**	{	}	[	]	**$**
$S \rightarrow$													
$T \rightarrow$													

(7)

	a	=	;	**if**	(	)	**else**	**while**	{	}	[	]	**$**
$E \rightarrow$													
$E' \rightarrow$													

(8)

III. (5 × 4)

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

授業・テストの感想