

I 拡張 BNF 記法で書かれた次の文法について考える。

$$\begin{aligned} \langle V \rangle &::= \text{"P"} \mid \text{"Q"} \mid \text{"R"} . & \langle L \rangle &::= [ \text{"!"} ] \langle V \rangle . \\ \langle P \rangle &::= \text{"1"} \mid \langle L \rangle \{ \langle L \rangle \} . & \langle S \rangle &::= \text{"0"} \mid \{ \langle P \rangle \text{"+"} \} \langle P \rangle . \end{aligned}$$

(1) 次の表中の終端記号列が、非終端記号  $L$ 、 $P$ 、 $S$  から導出できるかどうかについて、次の表の空欄に、導出できる場合は  $\square$  を、できない場合は  $\times$  をそれぞれ書き込みなさい。(12点)

終端記号列	$L$	$P$	$S$
0			
1			
P			
P1			
0+1			
P!Q!R			
!PP+!QQ+!RR			

(2) 終端記号列  $P!Q+R+1$  の非終端記号  $S$  に対する構文木を書きなさい。そのような構文木が存在しない場合は「構文木なし」と書きなさい。(4点)

学籍番号

氏名

(裏面に続く)

この試験の採点結果とこの科目の総合成績は、来週末までに Email で [txxxxxx@mail.ryukoku.ac.jp](mailto:txxxxxx@mail.ryukoku.ac.jp) 宛にお送りします。この連絡が不要な人は右の「連絡不要」を丸で囲んでください。

連絡不要

(3) 非終端記号  $P$  の導出規則を構文図で書き表しなさい。(4点)

(4) 非終端記号  $S$  の導出規則を構文図で書き表しなさい。(4点)

(5) (4) の構文図を決定的なものに書き換えなさい。(4点)

- (6) 以下の C プログラムは、標準入力(キーボード)から、文字列と改行文字を入力すると、入力された文字列が非終端記号  $S$  から導出できるかどうかを判定するプログラムの一部である。関数  $P$ 、 $L$ 、 $S$  の定義を補って、このプログラムを完成しなさい。(25 点)

```
#include <stdio.h>
#include <stdlib.h>

extern void V(void);
extern void P(void);
extern void L(void);
extern void S(void);

int t;

void error(void) {
    printf("Error!\n");
    exit(1);
}

void gettoken(void) {
    t = getchar();
}
```

```
void V(void) {
    switch (t) {
        case 'P': case 'Q': case 'R':
            gettoken();
            break;
        default:
            error();
    }
}

int main(void) {
    gettoken();
    S();
    if (t != '\n')
        error();
    printf("OK!\n");
    return 0;
}
```

```
void P(void)
{
```

```
}
```

(問題 I (6) の解答欄の続き)

```
void L(void)
{
```

```
}
```

```
void S(void)
{
```

```
}
```

(次ページに問題 II)

II 次は Minimum C のソースプログラムと、それをコンパイルして得られた MVM の機械語プログラムである。

### ソースプログラム

```

sqrt(n)
{
    int a, b;

    a = 0;
    b = n + 1;
    while (  ) {
        int t;

        t = ;
        if (  )
            b = t;
        else
            a = t;
    }
    return a;
}

int x, y;

main()
{
    input x;
    input y;

    while (x <= y) {
        print sqrt(x);
        x = x + 1;
    }
}

```

### コンパイル結果

```

0  ADD R13,R14,R0
4  LDI R11,272
8  CALL R0, 
12 EXIT R1
16 PUSH R13
20 ADD R13,R14,R0
24 SUBI R14,R14,8
28 LDI R1,0
32 ST R1,R13,-4
36 
40 
44 ADD R1,R1,R2
48 ST R1,R13,-8
52 LD R1,R13,-4
56 LDI R2,1
60 ADD R1,R1,R2
64 LD R2,R13,-8

```

```

68 SUB R0,R1,R2
72 JPGE R0,156
76 
80 LD R1,R13,-4
84 LD R2,R13,-8
88 ADD R1,R1,R2
92 LDI R2,2
96 DIV R1,R1,R2
100 ST R1,R13,-12
104 LD R1,R13,-12
108 LD R2,R13,-12
112 MUL R1,R1,R2
116 LD R2,R13,8
120 SUB R0,R1,R2
124 JPLE R0,140
128 LD R1,R13,-12
132 ST R1,R13,-8
136 
140 LD R1,R13,-12
144 ST R1,R13,-4
148 
152 JP R0,52
156 
160 ADD R14,R13,R0
164 POP R13
168 POP R15
172 ADDI R14,R14,8
176 POP R13
180 POP R15
184 PUSH R13
188 ADD R13,R14,R0
192 IN R1
196 ST R1,R11,0
200 IN R1
204 ST R1,R11,4
208 LD R1,R11,0
212 LD R2,R11,4
216 SUB R0,R1,R2
220 
224 LD R1,R11,0
228 PUSH R1
232 
236 ADDI R14,R14,4
240 OUT R1
244 LD R1,R11,0
248 LDI R2,1
252 ADD R1,R1,R2
256 
260 
264 POP R13
268 POP R15

```

- (1) ソースプログラムとコンパイル結果が対応するように、空欄部分を補いなさい。(30点)
- (2) この機械語プログラムを起動して、キーボードから整数 5 と 10 を順に入力したとする。120 番地の機械語命令が初めて実行される時(直前)の MVM のスタック中のデータ(整数値)を下図の空欄に書き込みなさい。ただし、下図の 1 つの欄は、それぞれ 32bit (4byte) のデータを表すものとし、プログラムの実行開始時には、スタックポインタ (R14) は  $2^{20}$  (= 1048576) で初期化されるものとする。スタックとして使用されていない領域は空欄のままにしておくこと。(12点)

MVM のメモリ (スタック) の内容

(アドレス)	⋮
1048532	
1048536	
1048540	
1048544	
1048548	
1048552	
1048556	
1048560	
1048564	
1048568	1048576
1048572	12
1048576	

- (3) (2) の時点での R1、R2、R11、R13 (フレームポインタ)、R14 (スタックポインタ) の値を下の欄に書き込みなさい。(5点)

R1	<input type="text"/>
R2	<input type="text"/>
R11	<input type="text"/>
R13	<input type="text"/>
R14	<input type="text"/>