Compiler

2019 Fall Middle Examination

Name____________ Student No.______________ Score__________

Problem 1: (40 points)

1.

(a) decimal = [0-9]|[1-9][0-9]+

(b) octal = 0[0-7]+

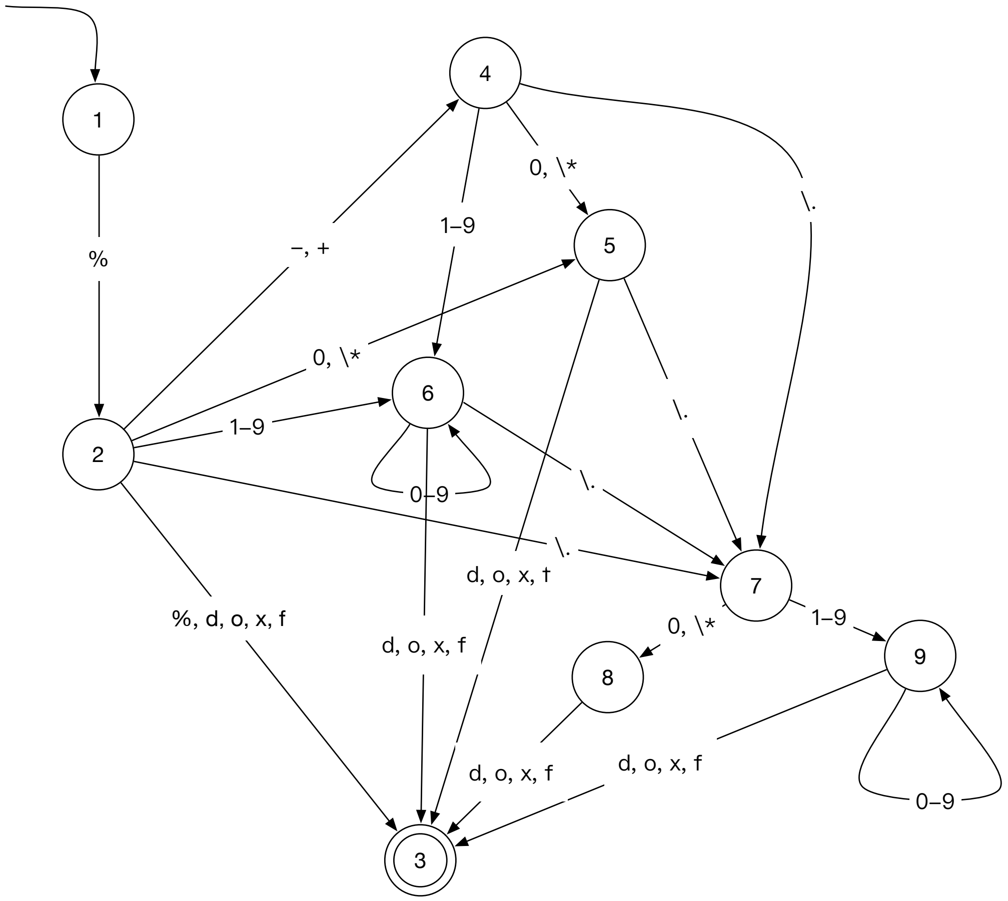
(c) hex = 0[xX][0-9a-fA-F]+

(d) formatString =

\% [-+]? (([0-9]|[1-9][0-9]+)|*)? (\.([0-9]|[1-9][0-9]+))? [doxf]|\%\%

NOTE: The answer is not unique.

2.



3.

```
int ==> 777

a ==> 7

= ==> 7

3 ==> 6

; ==> 2

printf ==> 1

( ==> 3

The formatted digit is ==> 9999

%+3.4x ==> 8

, ==> 5

0xfff ==> 6

) ==> 4

; ==> 2


7777762139999985642
```

Or:

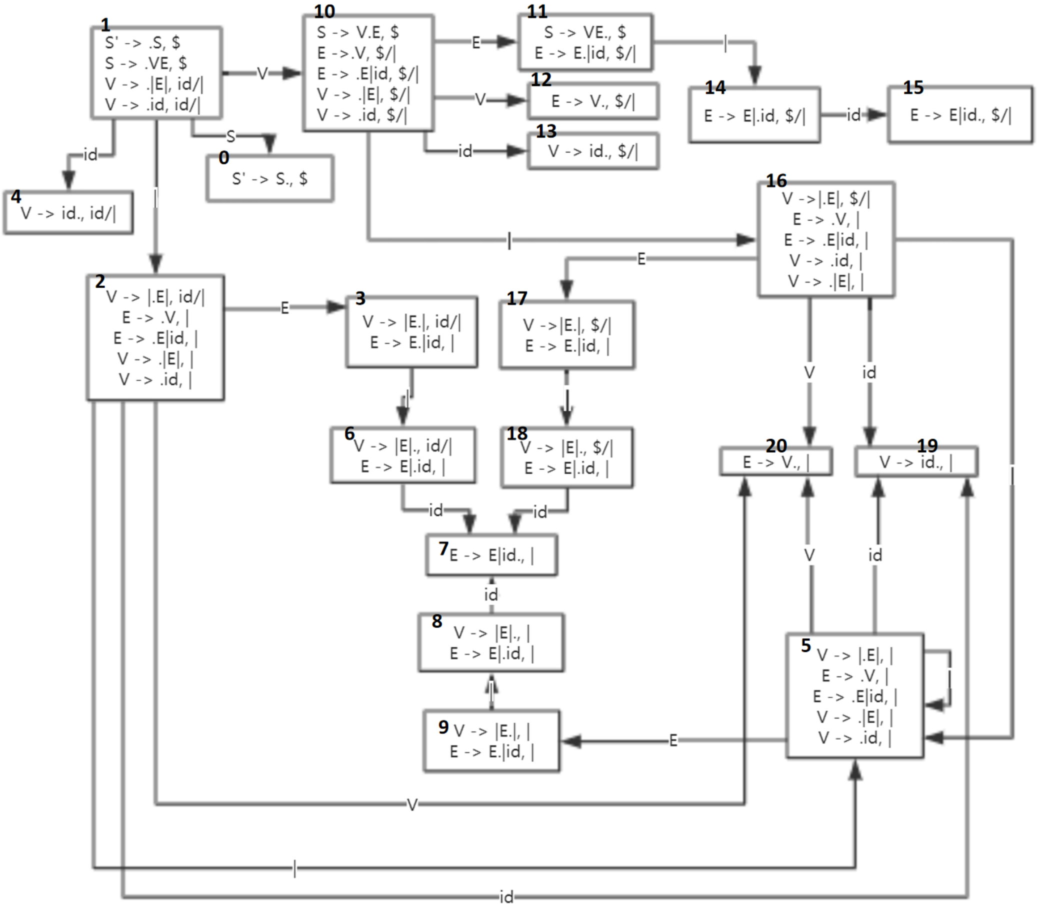
```
int ==> 777

a ==> 7

= ==> 7

3 ==> 6

; ==> 2

printf ==> 1

( ==> 3

The formatted digit is ==> 9999

%+3.4x ==> 8

\ ==> 10

n ==> 9

, ==> 5

0xfff ==> 6
) ==> 4
; ==> 2
```

```
777776213999981095642
```

实际上第二种答案才是正确的输出，但有很多同学将**\n**理解成了换行符，实际上**printf("The formatted digit is %+3.4x\n", 0xfff);**里的**\n**只是一个**'\'**字符以及一个**'n'**字符，这里并不是考点，因此两种情况都给分。

Problem 2: (60 points)

1.



It’s not a LR(1) grammar, because there is a S/R conflict in DFA.

2.

	$	|	id	V	E	S
0	R1/ Accept					
1		S2	S4	G10		G0
2		S5	S19	G20	G3	
3		S6				
4		R6	R6			
5		S5	S19	G20	G9	
6		R5	S7/R5			
7		R4				
8		R5	S7			
9		S8				
10		S16	S13	G12	G11	
11	R2	S14				
12	R3	R3				
13	R6	R6				
14			S15			
15	R4	R4				
16		S5	S19	G20	G17	
17		S18				
18	R5	R5	S7			
19		R6				
20		R4				
21						
22						
23						
24						
25						
26						
27						

1 shift/reduce conflict, 0 reduce/reduce conflict.

3.

When you encounter any conflict, list you choices here:

State6 S/R conflict on “id”, choose reduce/shift

choose reduce:

stack	action
$_1\vert_2$	Shift
$_1\vert_2 id_{19}$	Shift
$_1\vert_2 V_{20}$	Reduce6
$_1\vert_2 E_3$	Reduce3
$_1\vert_2 E_3\vert_6$	Shift
$_1 V_{10}$	Reduce5
$_1 V_{10} id_{13}$	Shift
$_1 V_{10} V_{12}$	Reduce6
$_1 V_{10} E_{11}$	Reduce3
$_1 V_{10} E_{11}\vert_{14}$	Shift
$_1 V_{10} E_{11}\vert_{14} id_{15}$	Shift
$_1 V_{10} E_{11}$	Reduce4
$_1 V_{10} E_{11}\vert_{14}$	Shift
$_1 V_{10} E_{11}\vert_{14} id_{15}$	Shift
$_1 V_{10} E_{11}$	Reduce4
$_1 S_0$	Reduce2
Accept	Reduce1

choose shift:

stack	action
1|2	Shift
1|2id19	Shift
1|2V20	Reduce6
1|2E3	Reduce3
1|2E3|6	Shift
1|2E3|6id7	Shift
1|2E3	Reduce4
1|2E3|6	Shift
1|2E3|6id7	Shift
1|2E3	Reduce4
1|2E3|6	Shift
1|2E3|6id7	Shift
Error	Error

Problem 1: Lexical Analysis (40 points)

```
/* lex definitions */
decimal             __________________  (Problem 1.1.1)
octal                __________________  (Problem 1.1.2)
hex                 __________________  (Problem 1.1.3)
float               decimal"."[0-9]*
number              decimal | octal | hex | float
formatString       __________________  (Problem 1.1.4)


/* start conditions */
%Start INITIAL STRING


%%
/* regular expressions and actions */
<INITIAL>(" " | "\n")                 {return SPACE;}
<INITIAL>printf                       {print("1"); return KEYWORD;}
<INTIAL>";"                           {print("2"); return SEMICOLON;}
<INTIAL>"("                           {print("3"); return LPAREN;}
<INTIAL>")"                           {print("4"); return RPAREN;}
<INITIAL>","                          {print("5"); return COMMA;}
<INITIAL>{number}                     {print("6"); return NUMBER;}
<INITIAL>\"                           {BEGIN(STRING);}
<INITIAL>.                            {print("7"); return ERROR;}
<STRING>(" " | "\n")                  {return SPACE;}
<STRING>{formatString}                {print("8"); return FORMATSTR;}
<STRING>[a-zA-Z0-9]+                  {print("9"); return WORD;}
<STRING>\"                            {BEGIN(INITIAL);}
<STRING>.                             {print("10"); return ERROR;}
```

In this problem, let’s consider an interesting lexical analyzer, which is able to analyze the contents of a **format string**, instead of simply considering them as a normal string. Format string is the first parameter of the **printf** function in C language E.g. **prinft("%2.3x", 20)**. "%2.3x" is the format string. With the related lex code listed above, please answer the following questions.

1. The first step of lexical analysis is specifying lexical structure using regular expressions. Please write the regular expressions for octal, hex and, formatString. Their descriptions are follow: (15')

   (a) Decimal numbers are base 10 because it have 10 different digits (from the 0 to the 9).

   (b) Octal numbers are the numbers based in 8, including only the representations for the values from 0 to 7(01234567). They are denoted by always beginning with a 0 digit.

   (c) Hexadecimal numbers have 16 different digits, that are represented by the numbers from 0 to 9 and the letters A(a), B(b), C(c), D(d), E(e) and F(f), which together serve us to represent the 16 different symbols that we need to express base 16 numbers. They are preceded by 0X or 0x.

   (d) The formatString here is a simple version of C language's format string. A format string follows this prototype:

   %[flags][width][.precision] specifier

   [flags]: Optinal. The valid flags are '-' and '+'.

   [width]: Optinal. The valid widths are decimal numbers or '*'.

   [.precision]: Optinal. The valid precisions are '.' followed by decimal numbers or '*'.

   specifier: Required. Specifier defines the type and the interpretation of its corresponding argument. The valid specifiers are **'d', 'o', 'x', 'f', and %. Different from C format string, when the specifier is %, the [flags][width][.precision] must all be empty.**

Valid examples:

0(decimal) 6(decimal) 23(decimal) 00(octal) 000(octal)

07(octal) 0x0(hex) 0X4Af(hex)

%+3.3x(formatString) %%(formatString) %0.4f(formatString)

Invalid examples:

08 (8 is not valid octal digit)

0xH (H is not valid hex digit)

0x (hex numbers must have at least 1 digits)

%.3% (The rest parts except % specifier are not empty)

2. Draw the **minimized** DFA that accepts **format strings**. For any regular expression, the minimized DFA is a unique DFA having the smallest number of states that accepts it. You will get part of scores if your DFA is not minimized. (20’)

3. What will be the output on the following input? The input is as follow: (5’)

```
int a = 3;
printf("The formatted digit is %+3.4x\n", 0xfff);
```

Problem 2: Grammar (60 points)

The following is a context-free grammar for a simple language, here are the rules for the grammar:

1. S′ → S$
2. S → V E
3. E → V
4. E → E|id
5. V → |E|
6. V → id

Using the above grammar, answer the following questions:

1. Construct the state graph (DFA) for this grammar using **LR(1)** items, do you think it’s a LR(1) grammar? (20’)

2. Follow the LR(1) procedure to construct the parsing table for the above DFA? How many **shift/reduce** conflicts in the table? How many **reduce/reduce** conflicts?(20’)

3. Use the parsing table to operate on the input string **“|id|id|id|id”**, show your parsing process. (20’)

   Notice:

   1). You can merge multiple continual shifts or reduces into one. E.g. for three shifts, write as “S 3”; for reduce 1, reduce 2, reduce 3, write as “R 123”).

   2). During parsing, when you encounter any conflicts, **please show you choices on each conflict before the process table**.