มหาวิทยาลัยสงขลานครินทร์

คณะวิศวกรรมศาสตร์

การสอบกลางภาค ประจำภาคการศึกษาที่ :1 **ประจำปีการศึกษา: 2548**

วันที่: 31 กรกฎาคม 2548 **เวลา:** 9.00-12.00

วิชา: 240-340 Compiler Structures **ห้อง:** A401

ทุจริตในการสอบ โทษขั้นต่ำคือ ปรับตกในรายวิชาที่ทุจริต และพักการเรียน 1 ภาคการศึกษา

คำสั่ง: อ่านรายละเอียดของข้อสอบ และคำแนะนำให้เข้าใจก่อนเริ่มทำข้อสอบ

อนุญาต: เครื่องเขียนต่างๆ เช่น ปากกา หรือดินสอ

ไม่อนุญาต: หนังสือ, เอกสารใดๆ และเครื่องคิดเลข

เวลา: 3 ชั่วโมง (180 นาที)

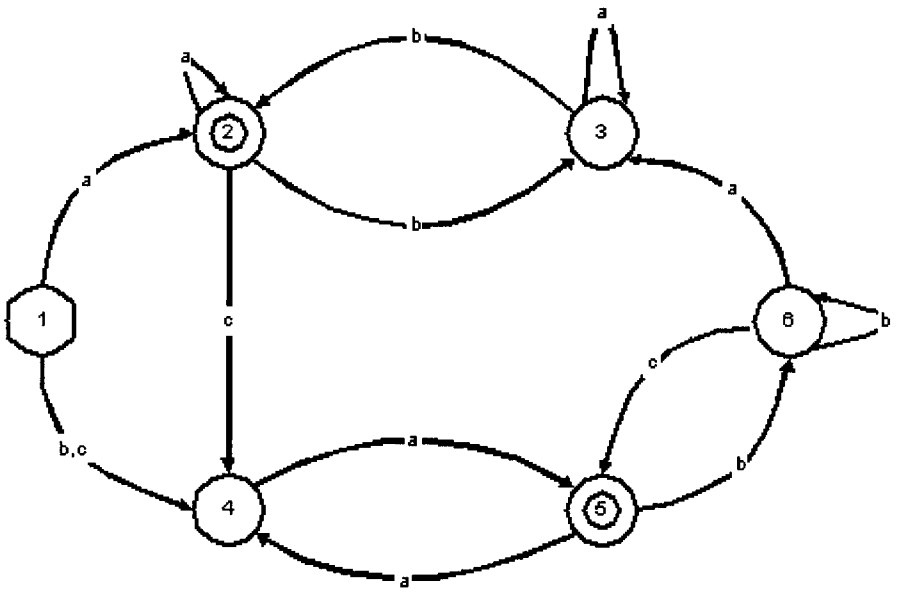
ชื่อ_____รหัสนักศึกษา_____Section _____

Notational conventions:

- non-terminal symbols of a grammar are in capital letters surrounded by angle brackets; example: `<TERM>`
- terminal symbols (tokens) are in lower-case letters surrounded by angle brackets, or are given literally in double quotes; examples: `"!"` `<number>` `"<="`
- the symbol `"::="` separates the left and right hand sides of a production
- the symbol `"e"` represents the null or empty string of zero characters

Computer Engineering 240-340
Compiler Structures
Midterm Examination, Term 1, 2547
Prince of Songkla University, Phuket/Had Yai

1. Consider the following finite state automaton over the input alphabet Σ of {a, b, c}..



a) Show the transition table for this FSA.

State	Input Symbol		
	A	b	c
1			
2			
3			
4			
5			
6			

b) Will the input

acabab

be accepted or rejected? Show all your work, giving the input and new state for each transition.

c) Will the input

```
ababacabc
```

be accepted or rejected? Show all your work, giving the input and new state for each transition.

d) Will the input

```
babacbaa
```

be accepted or rejected? Show all your work, giving the input and new state for each transition.

e) Will the input

```
caaab
```

be accepted or rejected? Show all your work, giving the input and new state for each transition.

2. Show regular expressions for the following descriptions. You may use concatenation, the alternation operator |, the * operator, and parentheses.

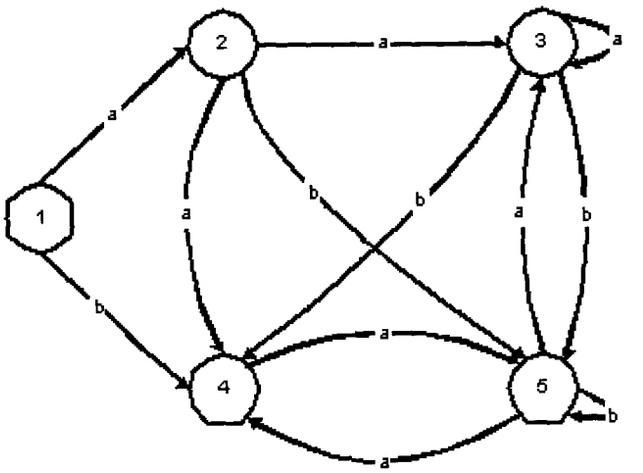
a) Two "a" characters, followed by at least two "b" characters, followed by an even number of "c" characters. The input alphabet Σ is {a, b, c}.

b) At least one binary digit, followed by a period character, followed by one or more additional binary digits. The input alphabet Σ is {0, 1, .}.

a) Either the letter "a" or the letter "b", followed optionally by any number of the letter "c", followed by an even number of the letter "b", followed by at least three copies of the sequence "ab". The input alphabet Σ is {a, b, c}.

3. Draw a finite state machine that accepts strings that begin with an "a" character, and then a "b" character or a "c" character, and which end with an "a" character. The input alphabet Σ is {a, b, c}.

4. Why is the following finite state machine non-deterministic? List all the problems with it.



a)

b)

c)

5. Mark with "yes" or "no" whether the following sets of productions are immediately left recursive and/or right recursive.

left recursive?	right recursive?	production
		<A> ::= <A> <a> <A> ::= <B>
		<B> ::= <a> <b> <B> ::= <C> <B> <a>
		<D> ::= <A> <a> <D> ::= <b> <B> <c> <D> ::= <a> <b> <D> <D> ::= <E>
		<E> ::= <A> <f> <E> <E> ::= <E> <a> <b> <E> ::= e
		<F> ::= <A> <F> <d> <F> ::= <a>

6. Consider the grammar:

```
(1)   <A> ::= <A> <d>
(2)   <A> ::= <g> <h>
(3)   <A> ::= <C> <k>
(4)   <B> ::= <C> <m>
(5)   <B> ::= <g>
(6)   <C> ::= <A> <n>
(7)   <C> ::= <B> <h>
```

a) Eliminate all left recursion. Show your work.

b) Left factor the resulting grammar to make it suitable for predictive parsing. Show your work.

7. Describe the kind of information that goes in each of the 3 parts of a lex input file.

a)

b)

c)

8. The following tokens are used by a certain lexical analyzer:

- the punctuation marks: `"&&"` `"=="` `">="` `">"` `"<="` `"<"` `"!="` `"||"` `"."`
- positive integer numbers consisting of one or more digital characters from `"0"` to `"9"`

Spaces, tabs and new-lines are to be ignored.

a) *flex* imposes ordering restrictions in the above token set. For each such restriction, which token must come before which other token(s)?

b) Show the contents of a C language header file that defines symbolic names for numerical token numbers, for the above tokens. Remember that the special end-of-file token must have the value 0.

b) Show the contents of the middle (second) part of a lex input file for processing the above tokens.

9. A certain grammar using the tokens of the previous question:

```
(1)  <PROGRAM> ::= <BOOLEAN> "."
(2)  <BOOLEAN> ::= <BOOLEAN> "&&" <OR>
(3)  <BOOLEAN> ::= <OR>
(4)  <OR> ::= <OR> "||" <RELATION>
(5)  <OR> ::= <RELATION>
(6)  <RELATION> ::= <number> "<" <number>
(7)  <RELATION> ::= <number> "<=" <number>
(8)  <RELATION> ::= <number> ">" <number>
(9)  <RELATION> ::= <number> ">=" <number>
(10) <RELATION> ::= <number> "==" <number>
(11) <RELATION> ::= <number> "!=" <number>
```

The implementation of the <PROGRAM> production should print the line "true" if its <BOOLEAN> right hand side term evaluates to logical true, and print the line "false" otherwise.

a) Eliminate left recursion in this grammar. Show all your work, step by step.

b)What things should the main program do to start parsing its input?

c)What should this main program not do at the start, because this grammar has no identifers?

d) What kind of error check should the main program make when parsing is finished?

e) Draw the parser tree using the original grammar for:

f) If the above parser is run with this input:

```
3 > 6 && 2 < 1 || 5 <= 7 .
```

what output is printed?

g) How does this differ from the usual evaluation of such expressions in most programming languages?

h) Write C code for a method that implements the production for `<PROGRAM>`, as you did in the lab exercise. Assume that there is an `int` global variable called *`lookahead`*, and a `char * ` global variable called *`yytext`*, that are updated by the lexical analyzer and a `match(int)` utility method. Show the processing of terminal and non-terminal symbols; if the method is not void, show how its return value is calculated. Refer to the token names in the header file, not the absolute token numbers.

i) Write C code as in the previous item, for a method that implements the <OR> production of your transformed grammar that eliminates left recursion.

j) Write C code as in the previous item, for a method that implements the new non-terminal symbol that goes with the <OR> production of your transformed grammar that eliminates left recursion.