1. [10] Eliminate left recursion from the grammar

   S → Aa | b

   A → Sc | Ad | e

2. Consider the following grammar G:

   $E' \rightarrow E$

   $E \rightarrow (L) \mid a$

   $L \rightarrow E\ L \mid E$

   (a) [10] Construct the collection of the sets of LR(0) items

   (b) [10] When constructing the action table of SLR parser of G, what are the rules to determine the parsing actions? That is, what is the rule for a shift action at state *i*? What is the rule for a reduce action at state *i*?

   (c) [10] Construct the SLR parsing table of *G*. Please specify clearly how every shift or reduce action is determined

3. Consider the following grammar G

   $S \rightarrow ABd$

   $A \rightarrow aA \mid \epsilon$

   $B \rightarrow b \mid cA$

   (a) [5] What is the language generated by G

   (b) [10] Write down the FIRST and FOLLOW sets for all nonterminals of G

   (c) [10] Show the predictive parsing table of G

4. Consider the following grammar G

   S → iEtS | iEtSeS | a

   E → b

   where S and E are nonterminals and i, t, e, a, and b are terminals

   (a) [10] Is G LL(1)? If yes, why? If not, why?

   (b) [10] Is G SLR(1)? If yes, why? If not, why?

5. Consider the grammar G

   $S' \rightarrow S$

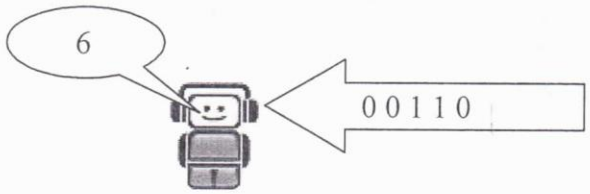
   $S \rightarrow L = R \mid R$

   $L \rightarrow * R \mid \mathbf{id}$

   $R \rightarrow L$

   (a) [10] Build the goto graph of the sets of LR(1) items

   (b) [10] Construct the LALR(1) parsing table of *G*



6. iRobot only knows binary digits: **0** and **1**. After he hears a sequence of binary digits, he will convert the binary number into a decimal value. For example, it computes the decimal value of the binary number 0 0 1 1 0 is 6.

Our goal is to use LEX and YACC to write a compiler that will perform the following tasks:

(1) scan and parse the binary digits,
(2) generate a Java bytecode program that computes the decimal value, and
(3) print iRobot's answer.

(a) [15] Write a LEX program and a YACC program that will work together to scan and parse iRobot's input.
(b) [10] Extend your YACC program so that it will generate a sequence of Java bycode instructions that convert a binary number into its decimal value.
(c) [10] Use your compiler to translate the sequence binary digits 0 0 1 1 0 into a Java bytecode program.

For your reference,

PRINT Statements *print expression;*

The PRINT statements in *sC* are modeled by invoking the *print* method in *java.io* package using the following format

```
getstatic java.io.PrintStream java.lang.System.out
...  /* compute expression */
invokevirtual void java.io.PrintStream.print(java.lang.String)
```

if the type of *expression* is a string. Types *int* or *boolean* will replace *java.lang.String* if the type of *expression* is integer or boolean.