CSE-304 Compiler Design

Fall 2000

Mid-term Exam

Oct 31, 2000.

Duration: 1 hour, 10 minutes.

ID Number : _____

Name : _____

INSTRUCTIONS

Read the following carefully before answering any question.

- MAKE SURE YOU HAVE FILLED IN YOUR NAME AND ID NUMBER IN THE SPACE ABOVE.
- The exam for undergraduate students is different from the exam for the graduate students. Make sure you are reading the correct question paper.
- Doubts about the questions will be answered only in the first 15 minutes of the exam. So, read the questions carefully at the beginning of the exam.
- Write your answers in the space provided.
- Keep your answers brief and precise.
- The exam consists of 5 questions, in 8 pages (including this page) for a total of **110** points.
  You may attempt all questions. However, the maximum you can score in this exam is **100** points.

GOOD LUCK!

Question	Points
1.	
2.	
3.	
4.	
5.	
Total	

1. **[Total: 20 points]** Consider the following two languages:

   L_1: strings over $\{a, b\}$ such that every a is immediately followed by ***at least one*** b.

   L_2: strings over $\{a, b\}$ such that every a is immediately followed by ***at most one*** b.

   L_3: strings over $\{a, b\}$ such that the number of a's is same as the number of b's.

   a. (7 points) Is L_1 a regular language? If so, write a regular expression corresponding to L_1. If not, explain why not.

   b. (7 points) Is L_2 a regular language? If so, write a regular expression corresponding to L_2. If not, explain why not.

   b. (6 points) Is L_3 a regular language? If so, write a regular expression corresponding to L_3. If not, explain why not.

2. **[20 points]** Consider text files containing financial figures: english sentences, punctuation symbols, and numbers preceded by/followed by currency symbols. For instance,

```
Airfare:       $454.72
Europass:       207.00DM
Hotel:          127DM
Cab to Airport: $30.00
Going to Europe in the middle of the semester (Oct 31): priceless!
```

is such a file. In this file "`$`" and "`DM`" stand for the currencies US Dollar and German Mark respectively.

Your task is to write a currency converter **as a lex program** for these files, changing every currency figure into Francs. Money in Francs is denoted by attaching "`Fr`" to the end of the number (as is the case with "`DM`"). Assume that all figures denoting money in the input file are either in US Dollars or in German Marks.

Numbers denoting money are written with an optional decimal point. Numbers without a currency symbol denote quantities other than money, and should be left unchanged. You may assume that any number representing money has exactly one currency symbol preceding/following it.

Use the rates \$1 = 7.73Fr and 1DM = 3.35Fr to convert the currencies. The rest of the file should remain unchanged. For instance, for the above example file, the output of your program should look like:

```
Airfare:       3514.99Fr
Europass:       693.45Fr
Hotel:          425.45Fr
Cab to Airport: 231.90Fr
Going to Europe in the middle of the semester(Oct 31): priceless!
```

3. **[Total: 25 points]** Consider the context-free grammar G_3:

$$\begin{aligned} S &\longrightarrow AaAb \\ S &\longrightarrow Bb \\ A &\longrightarrow \epsilon \\ B &\longrightarrow \epsilon \end{aligned}$$

a. (6 points)

$FIRST(A) =$

$FIRST(B) =$

$FIRST(S) =$

a. (6 points)

$FOLLOW(S) =$

$FOLLOW(A) =$

$FOLLOW(B) =$

c. (8 points) Is G_3 LL(1)? Justify.

4. **[Total: 25 points]** Give the SLR parsing table (action and goto table) for the following grammar (same as G_3):

$$
\begin{aligned}
S &\longrightarrow AaAb \\
S &\longrightarrow Bb \\
A &\longrightarrow \epsilon \\
B &\longrightarrow \epsilon
\end{aligned}
$$

Your work must show the item set construction (10 points); the SLR action table (Shift entries: 8 points, Reduce entries: 4 points); and the goto table (3 points).

5. **[Total 25 points]**Some langauges allow users to define new infix operators in their programs; then the users also specify the associativity and precedence of these operators.

   Assume that the new operators are defined using the syntax:

   `new_operator` *operator_id precedence associativity*

   where *operator_id* is a sequence of non-white-space characters; *precedence* is one of `high`, `medium` and `low`; and *associativity* is one of `left` and `right`. The *operator_id*, *precedence*, *associativity* fields are separated by one or more whitespace characters. For instance, new operators "`/\`" and "`\/`" which are both left-associative such that "`/\`" has higher precedence than "`\/`" can be defined using this syntax as: the usual definition of "*" and "+" w

   ```
   new_operator /\ high left
   new_operator \/ medium left
   ```

   a. (4 points) Consider an operator "`#`" defined over expressions e such that e_1`#`e_2`#`e_3 is equivalent to e_1`#`(e_2`#`e_3) and not same as (e_1`#`e_2)`#`e_3. Is "`#`" left-associative/right-associative/non-associative?

   b. (4 points) Consider an operator "`|`" defined over expressions e such that e_1`|`e_2`|`e_3 is equivalent to (e_1`|`e_2)`|`e_3 and not same as e_1`|`(e_2`|`e_3). Is "`|`" left-associative/right-associative/non-associative?

   c. (7 points) Using the `new_operator` construct described above, write the defintions of "`#`" and "`|`" given that e_1`|`e_2`#`e_3 is equivalent to e_1`|`(e_2`#`e_3) and not to (e_1`|`e_2)`#`e_3.

[Continued on next page]

d. (10 points) Describe how you will build a scanner and parser for a language that allows user-defined operators using Lex and Bison. You need not write Lex or Bison code. Describe how the grammar for expressions would look like, how you would encode this grammar in Bison, and how tokens for this grammar will be generated by a scanner built using Lex, and what the symbol tables would contain.

[Hint: With only 3 precedence levels and 2 associativity directions, six abstract operators can be used to represent the user-defined ones.]
