| | | |
|---|---|---|
| ***Ministry of Higher Education and Scientific Research*** | | |
| ***Salahaddin University / Erbil*** | | ***Subject: Compilers*** |
| ***College of Engineering*** | | ***Date: ---*** |
| ***Dept. of Software Engineering*** | ***Mid Year Exam*** | ***Time: 90 Minutes*** |
| | ***2011-2012*** | ***Lecturer: Amanj Sherwany*** |

*The highest obtainable mark is 17, the minimum passing mark is 8.5*

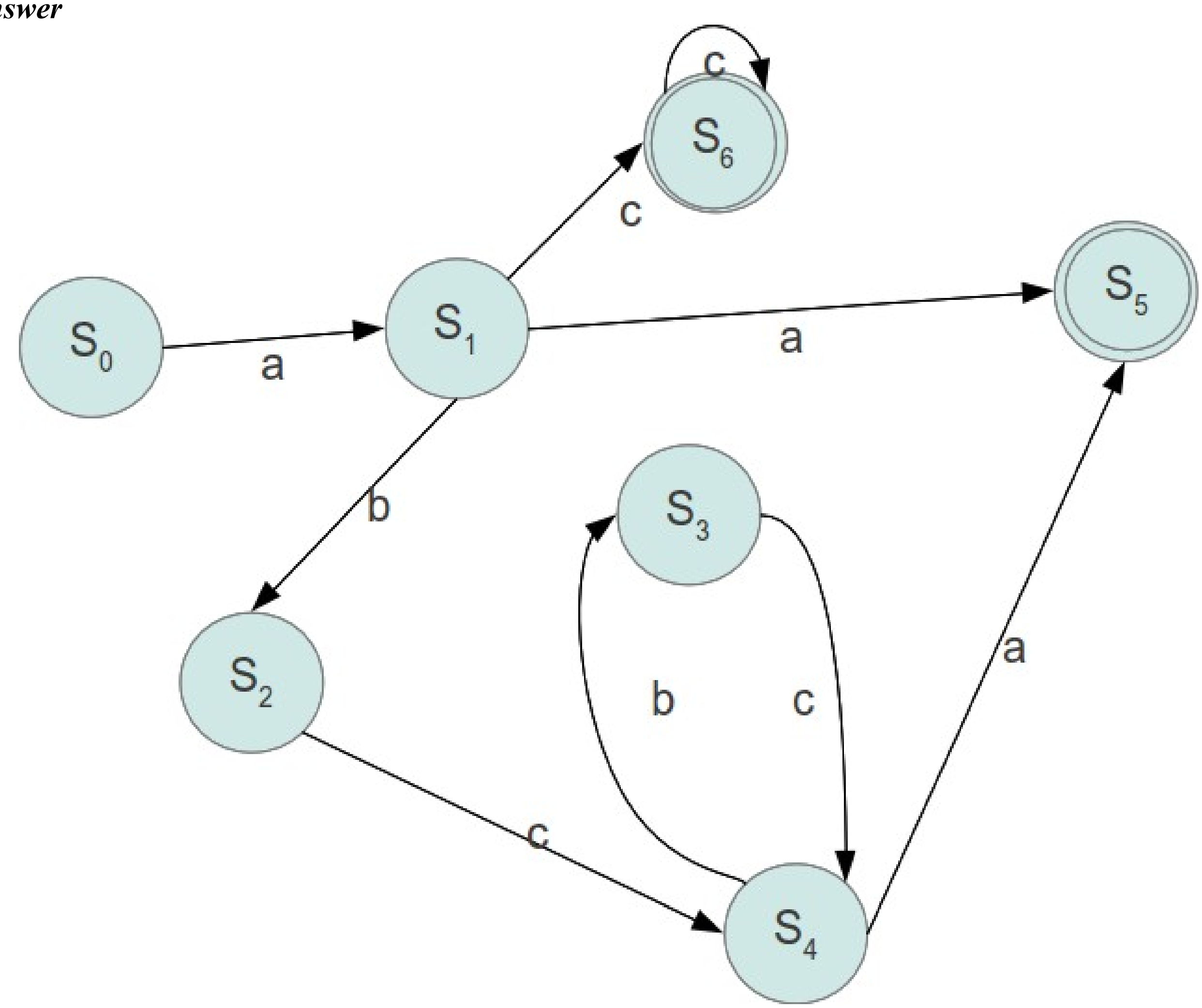
**Q1: (4 points)**

For the the following regular expression:

*a(bc)*a | ac(c)**

1. Construct a Deterministic Finite Automaton (DFA) for the above regular expression. **(3 points)**

***Answer***



2. Show each step of the lexer on the string *aaaccabca*. Be sure to show the values of the important internal variables of the recognizer. There will be repeated calls to the lexer to get all tokens from the string. **(1 point)**

***Answer***

| **Input String** | **Start Index** | **Current Index** | **Accepting Index** | **Token** |
|---|---|---|---|---|
| **a**aaccabca | 0 | 0 | | |
| **aa**accabca | 0 | 1 | 1 | aa |
| aa**a**ccabca | 2 | 2 | | |
| aa**ac**cabca | 2 | 3 | 3 | |
| aa**acc**abca | 2 | 4 | 4 | acc |
| aaacc**a**bca | 5 | 5 | | |
| aaacc**ab**ca | 5 | 6 | | |
| aaacc**abc**a | 5 | 7 | | |
| aaacc**abca** | 5 | 8 | 8 | abca |

*   *   *

**Q2: (4 points)**

Consider the following context-free grammar $G_0$:

$E \rightarrow E, E$
$E \rightarrow E = E$
$E \rightarrow E[E]$
$E \rightarrow (E)$
$E \rightarrow \mathrm{x}$

1. The grammar $G_0$ is ambiguous. Explain what this concept means, and give an example which shows that $G_0$ is ambiguous. **(2 points)**

***Answer***

*Ambiguous grammars are the kind of grammars which can generate two different parse trees for the same input string.*

*An example to show that the above grammar is ambiguous:*

*x=x,x*

*Which can be generated in two different ways:*

*a-* $E \rightarrow E, E \rightarrow E = E, E \rightarrow x = x, E \rightarrow x = x, x$

*b-* $E \rightarrow E = E \rightarrow x = E \rightarrow x = E, E \rightarrow x = x, E \rightarrow x = x, x$

2. Assume that ($E[E]$) has the highest priority, (=) has the second-highest priority, and (,) has the lowest priority. Rewrite $G_0$ to an equivalent $G_1$ which expresses these properties. **(1 point)**

***Answer***

$$E \rightarrow E, E \mid T$$
$$T \rightarrow T = T \mid F$$
$$F \rightarrow F[F] \mid K$$
$$K \rightarrow (E) \mid \mathrm{x}$$

3. Assume that (=) is right-associative and (,) is left-associative. Rewrite $G_1$ to an equivalent grammar $G_2$ which expresses these properties. **(1 point)**

***Answer***

$$E \rightarrow E, T \mid T$$
$$T \rightarrow F = T \mid F$$
$$F \rightarrow F[F] \mid K$$
$$K \rightarrow (E) \mid \mathrm{x}$$

* * *

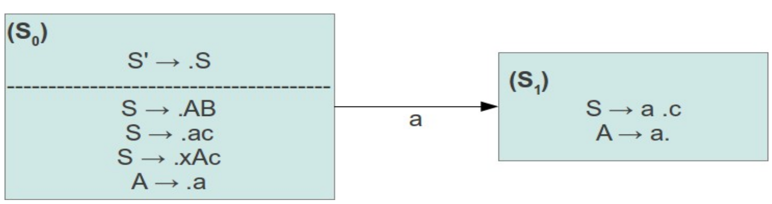
**Q3: (6 points)**

Consider the following augmented grammar, where $S'$ is the start symbol and \$ is the special end-of-input symbol.

$P_0 \quad S' \rightarrow S\$$
$P_1 \quad S \rightarrow A\ B$
$P_2 \quad S \rightarrow \mathrm{a\ c}$
$P_3 \quad S \rightarrow \mathrm{x}\ A\ \mathrm{c}$
$P_4 \quad A \rightarrow \mathrm{a}$
$P_5 \quad B \rightarrow \mathrm{b}$
$P_6 \quad B \rightarrow \text{-}$

1. Show that this grammar is not *SLR* (or, show that the *SLR* construction will fail). **(1 points)**

***Answer***




*The FOLLOW of A is {b, c} so the Action/Goto table of $S_1$ will be like:*

| | Action | | | | | | Goto | | |
|---|---|---|---|---|---|---|---|---|---|
| **States** | **a** | **c** | **x** | **b** | **-** | **\$** | **S** | **A** | **B** |
| $S_1$ | | s1, r4 | | r4 | | | | | |

*In state $S_1$ we have both shift and reduce, which means this is not an SLR grammar.*

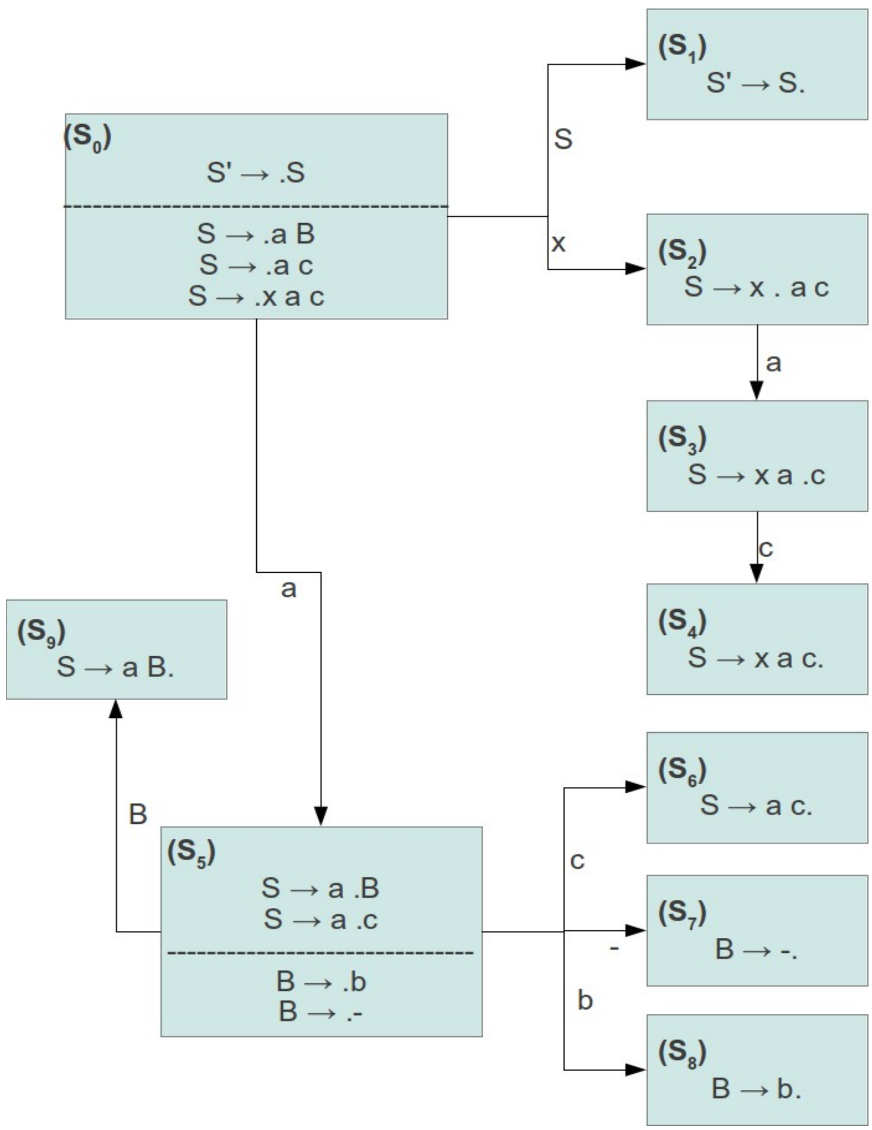
2. Rewrite the grammar to an equivalent grammar that is *SLR*. (Hint: clone or eliminate the non-terminal *A*.) Construct an *SLR* parsing table for the new grammar, including the intermediate *LR(0)* automation with states and transitions, and the *FIRST* and *FOLLOW* sets for the non-terminals. **(4 points)**

***Answer***

*By eliminating the terminal A we will get the following grammar:*

| | |
|---|---|
| $P_0$ | $S' \rightarrow S\$$ |
| $P_1$ | $S \rightarrow$ a $B$ |
| $P_2$ | $S \rightarrow$ a c |
| $P_3$ | $S \rightarrow$ x a c |
| $P_4$ | $B \rightarrow$ b |
| $P_5$ | $B \rightarrow$ - |

*The LR(0) DFA will look like:*



*The SLR table:*

| | Action | | | | | | Goto | |
|---|---|---|---|---|---|---|---|---|
| **States** | **a** | **c** | **x** | **b** | **-** | **$** | **S** | **B** |
| $S_0$ | s5 | | s2 | | | | g1 | |
| $S_1$ | | | | | | A | | |
| $S_2$ | s3 | | | | | | | |
| $S_3$ | | s4 | | | | | | |
| $S_4$ | | | | | | r3 | | |
| $S_5$ | | s6 | | s8 | s7 | | g9 | |
| $S_6$ | | | | | | r2 | | |
| $S_7$ | | | | | | r5 | | |
| $S_8$ | | | | | | r4 | | |
| $S_9$ | | | | | | r1 | | |

*FIRST(S) = {a, x}, FIRST(B) = {b, -}     FOLLOW(S)={$}, FOLLOW(B)={$}*

3. Show how an *LR* parser step by step (including changes in the stack and remaining tokens) parses the string *xac* using your *SLR* table. **(1 point)**

***Answer***

| Stack | Input | Action |
|---|---:|---|
| $0 | xac$ | s2: Shift on x |
| $0x2 | ac$ | s3: Shift on a |
| $0x2a3 | c$ | s4: Shift on c |
| $0x2a3c4 | $ | Reduce S → xac<br>Pop \|xac\|*2, exposing 0, push S<br>Goto(0, S)=1 |
| $0S1 | $ | Accept |

*     *     *

**Q 4: (3 points)**

Knowing that C-like languages use static scoping:

1. You are writing a C-compiler for a device that has a *2.4 GHz i7* CPU and only *256 MB* of RAM. Describe an *efficient* way of processing nested scopes for your compiler. Motivate your answer. **(2 points)**

*This is a device with a high speed and a low memory, so an efficient way for handling the nested scopes is:*

- *First traverse the scope's declarations (the syntax tree) and enter them in the hash table.*
- *Type-check the statements/expressions inside the scope.*
- *Before leaving the scope, traverse the scope's declarations again and remove them from the hash table.*

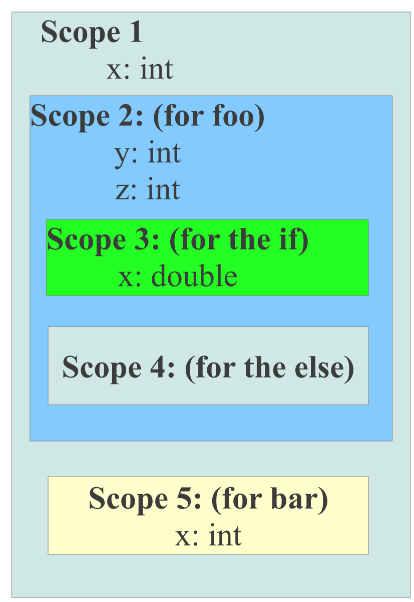
*since this alternative does not use an extra memory, although it is a bit slow.*

2. Identify the *scopes* and the *variable* declarations for each scope of the following C program. **(1 point)**

```
int x;
int foo(int y)
{
     int z;
     if (y != 0) {
          double x = 3.14;
          z = bar(x);
     } else
          z = x;
     return z;
}

int bar(void)
{
     int x = 2;
     return x;
}
```

***Answer***




Good Luck