$Id: cmps104a-2015q2-exam2.mm,v 1.76 2015-05-08 17:17:38-07 - - $

page 1 page 2 page 3 Total / 32

Please print clearly :

Name : SOLUTION

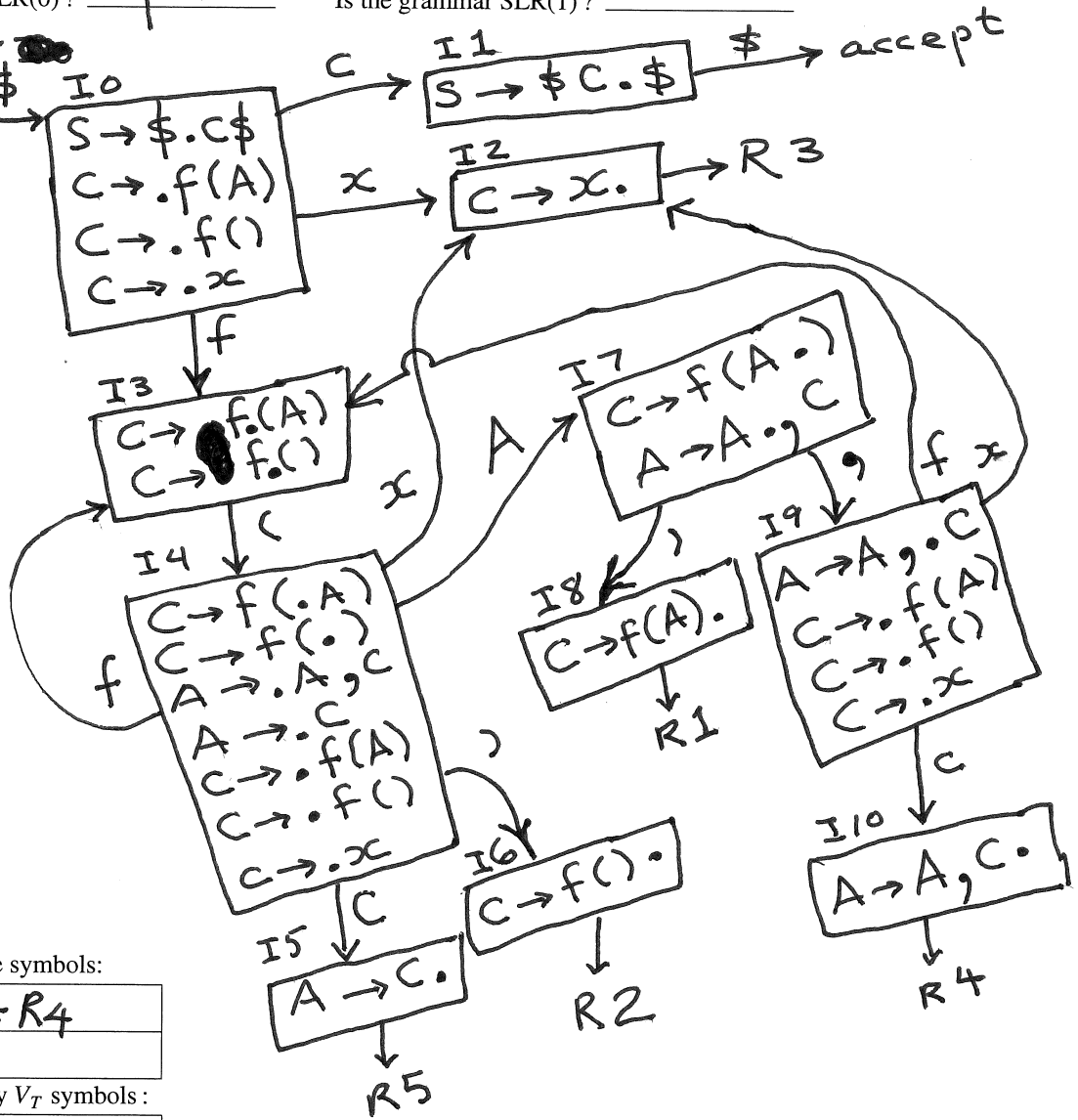
Login : @ucsc.edu

No books ; No calculator ; No computer ; No email ; No internet ; No notes ; No phone. Neatness counts ! Do your scratch work elsewhere and enter only your final answer into the spaces provided.

1. Given the grammar presented here, and using the style from the LALR(1) handout :
   (a) Construct the characteristic finite state machine (CFSM), sets of items and transition diagram, showing shifts, reductions, and acceptance. **[6✓]**
   (b) Construct the FOLLOW sets. Show the first pass with rule symbols in the Follow sets. Then show the revised follow sets with only terminal symbols. (See chart at the bottom of the page.) **[3✓]**
   (c) Answer ***yes*** or ***no*** to each of the following questions : **[1✓]**

   Is the grammar LR(0) ? yes Is the grammar SLR(1) ? yes

0.	S → $ C $
1.	C → f (A)
2.	C → f ( )
3.	C → x
4.	A → A , C
5.	A → C



Follow sets with rule symbols:

Follow(C) :	$ R5 R4
Follow(A) :	),

Follow sets with only V_T symbols :

Follow(C) :	$),
Follow(A) :	),

2. Write a `bison` grammar for the language described here, which is a small subset of `oc`. Assume the functions `adopt1` and `adopt2` from the project, and build the abstract syntax tree according to the project specs. Once the AST has been constructed, store the root in an external variable called `astroot`. Your grammar must be unambiguous. **[5✔]**
   (a) The start symbol is an `expr`.
   (b) An `expr` is one or more terms connected by `'+'` operators. The `'+'` operator is left associative.
   (c) A `term` is either an `ATOM` or an ATOM followed by an `arglist` inside parentheses.
   (d) An `arglist` is a sequence of zero or more `expr`s.
   (e) If an `arglist` has two or more `expr`s, each `expr` is separated from the next with a comma.

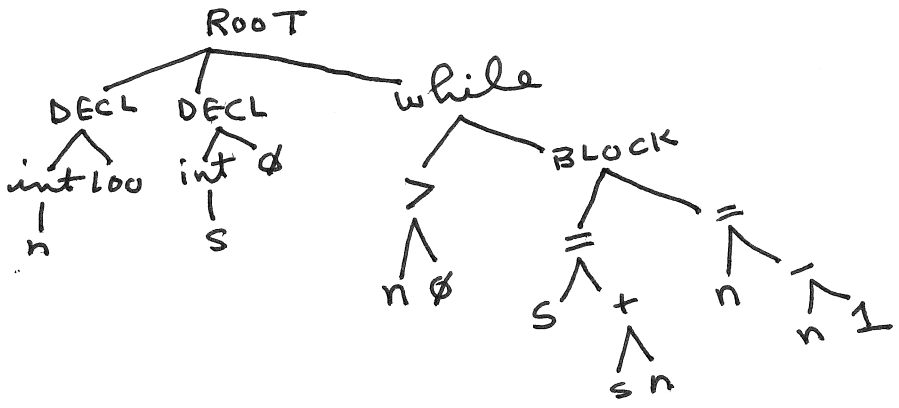
```
%start start
%token ATOM
%%
start    : expr                 { astroot = $1; }
         ;
expr     : expr '+' term        { $$ = adopt2 ($2, $1, $3); }
         | term                 { $$ = $1; }
         ;
term     : ATOM                 { $$ = $1; }
         | ATOM '(' ')'         { $$ = adopt1 ($2, $1); }
         | atomargs ')'         { $$ = $1; }
         ;
atomargs : ATOM '(' expr        { $$ = adopt2 ($2, $1, $3); }
         | atomargs ',' expr    { $$ = adopt1 ($1, $3); }
         ;
```

3. Write a `flex` grammar for the language described above. **[3✔]**
   (a) Call the function `yylval_token` as per the project whenever a token is recognized.
   (b) Comments consist of the Java-style //-comments.
   (c) White space consists of spaces, tabs, and newlines.
   (d) For any invalid character found, call the function `lexerror` with that character as its argument.

```
"//".*                  { }
[ \t\n]+                { }
[0-9A-Za-z]+            { return yylval_token (ATOM); }
"+"                     { return yylval_token ('+'); }
","                     { return yylval_token (','); }
"("                     { return yylval_token ('('); }
")"                     { return yylval_token (')'); }
.                       { lexerror (*yytext); }
```

4. Draw the abstract syntax tree according to the project specifications for the following program. **[2✔]**

```
int n = 100;
int s = 0;
while (n > 0) {
   s = s + n;
   n = n - 1;
}
```



Multiple choice. To the ***left*** of each question, write the letter that indicates your answer. Write **Z** if you don't want to risk a wrong answer. Wrong answers are worth negative points. **[12✔]**

number of correct answers		× 1 =	$= a$
number of wrong answers		× ½ =	$= b$
number of missing answers		× 0 =	0
column total $c = \max(a-b, 0)$	12		$= c$

1. A parser, such as is generated by `bison`, is a :
B
(A) finite state automaton
(B) pushdown automaton
(C) linear bounded Turing machine
(D) universal Turing machine

2. Which of the following grammars will recognize an x followed by zero or more occurrences of y ?
B
(A) $A \rightarrow A\ x \qquad A \rightarrow y$
(B) $A \rightarrow A\ y \qquad A \rightarrow x$
(C) $A \rightarrow x\ A \qquad A \rightarrow y$
(D) $A \rightarrow y\ A \qquad A \rightarrow x$

3. Which of the following might be a function prolog ?
A
(A) `push fp; fp = sp; sp -= N;`
(B) `push sp; sp = fp; fp -= N;`
(C) `sp -= N; push fp, sp = fp;`
(D) `sp = fp; push sp; sp -= N;`

4. Which of the following might occur in a function epilog ?
D
(A) `fp = sp; pop sp; return;`
(B) `pop fp; sp = fp; return;`
(C) `pop sp; fp = sp; return;`
(D) `sp = fp; pop fp; return;`

5. Assuming 64-bit pointers, and memory management using the boundary tag method, what is the closest distance between addresses that might be returned from successive calls to `malloc`(3) ?
C
(A) 0x08
(B) 0x10
(C) 0x20
(D) 0x40

6. Which of the following items in a state will cause a shift ?
B
(A) $E \rightarrow \bullet\ E + T$
(B) $E \rightarrow E \bullet + T$
(C) $E \rightarrow E + \bullet\ T$
(D) $E \rightarrow E + T \bullet$

7. Which of the following items in a state was added during the closure computation ?
A
(A) $E \rightarrow \bullet\ E + T$
(B) $E \rightarrow E \bullet + T$
(C) $E \rightarrow E + \bullet\ T$
(D) $E \rightarrow E + T \bullet$

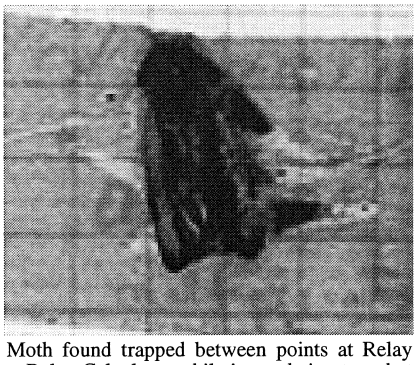
8. Which of the following is only needed if a language permits functions to be nested ?
D
(A) dynamic link
(B) frame pointer
(C) return address
(D) static link

9. Recognizing reserved words by coding them as regular expressions in the scanner will make the generated DFA :
D
(A) run more quickly.
(B) run more slowly.
(C) use less memory.
(D) use more memory.

10. Which statement will round `n` up to the next higher multiple of 16 if it is not a multiple of 16 and leave it unchanged if it is a multiple of 16 ?
B
(A) `n = (n + 15) & ! 15;`
(B) `n = (n + 15) & ~ 15;`
(C) `n = n + (15 & ~ 15);`
(D) `n = n >> 4 << 4;`

11. What class of languages is equivalent to the universal Turing machine ?
D
(A) regular languages
(B) context free languages
(C) context sensitive languages
(D) unrestricted languages

12. Who invented the Turing machine in 1936?
A
(A) Alan Turing
(B) Alonzo Turing
(C) Charles Turing
(D) Noam Turing



The First "Computer Bug". Moth found trapped between points at Relay #70, Panel F, of the Mark II Aiken Relay Calculator while it was being tested at Harvard University, 9 September 1947. The operators affixed the moth to the computer log, with the entry: "First actual case of bug being found." They put out the word that they had "debugged" the machine, thus introducing the term "debugging a computer program". In 1988, the log, with the moth still taped by the entry, was in the Naval Surface Warfare Center Computer Museum at Dahlgren, Virginia. [`http://en.wikipedia.org/wiki/File:H96566k.jpg`]