$Id: cmps104a-2015q4-exam3.mm,v 1.54 2015-12-01 16:29:47-08 - - $

page 1 page 2 page 3 page 4 page 5 Total / 54

Please print clearly :

Name : SOLUTION

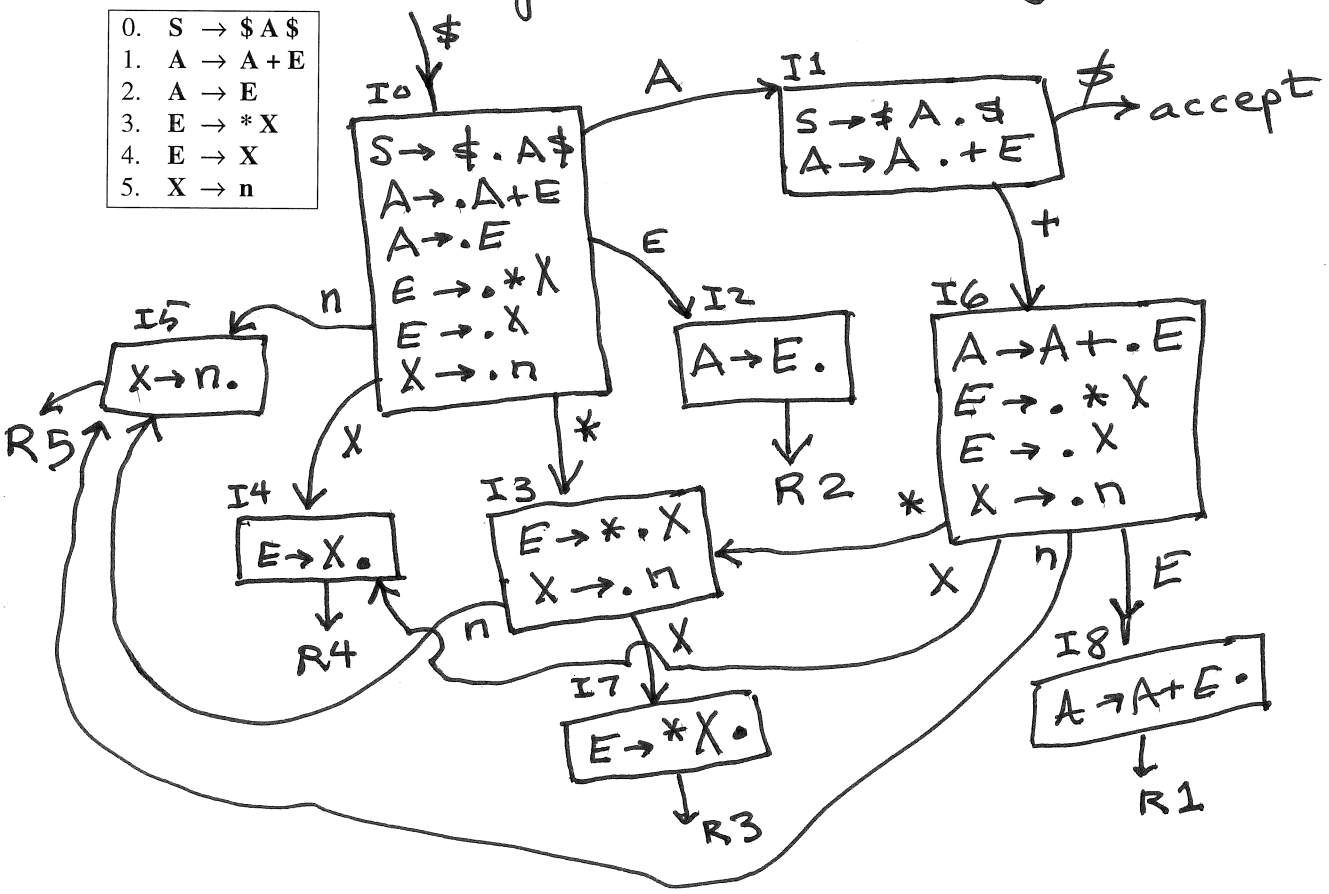
Login : @ucsc.edu

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 shift, reduce, and accept actions. **[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 → $ A $
1.	A → A + E
2.	A → E
3.	E → * X
4.	E → X
5.	X → n



Follow sets with rule symbols:

Follow(E) :	R2 R1
Follow(A) :	$ +
Follow(X) :	R4 R3

Follow sets with only V_T symbols:

Follow(E) :	$ +
Follow(A) :	$ +

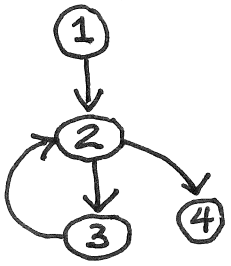
FOLLOW(X) : $ +

2. Basic blocks.
   (a) Given the code shown in the box, draw a horizontal line immediately above the leader of each basic block, thus separating it from the last instruction in the preceding basic block. Number the basic blocks in sequence as 1, 2, ..., etc. in the same order as the instructions appear. **[1✓]**
   (b) Draw a flow diagram with each circle in the diagram having the number of a basic block, and with arrow showing flow of control. **[1✓]**
   (c) Draw the dominator tree with solid arrows showing the **dom** relationship. Draw a dotted arrow showing the back edge. Write an asterisk next to the head of the natural loop. **[1✓]**

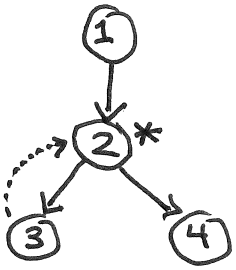
```
fac:    f = 1
        n = 1
do:     if (n > 5) goto od
        r1 = f
        r1 *= n
        f = r1
        r2 = n
        r2 += 1
        n = r1
        goto do
od:     call prt (f)
        return
```

(a) 1 2 3 4

(b)



(c)



3. Given a function prolog listed in the box at the left, write code using the `movq` and `popq` instructions for the epilog immediately before the `ret` instruction. **[1✓]**

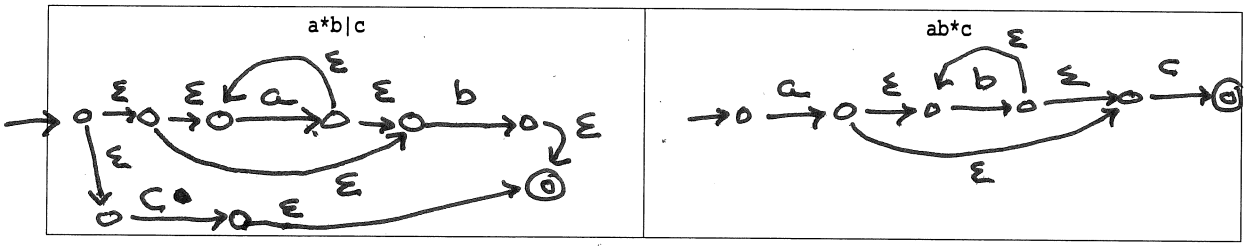
prolog:

```
pushq   %rbp
movq    %rsp, %rbp
subq    $80, %rsp
```

epilog:

```
movq %rbp, %rsp
popq %rbp
ret
```

4. Using Thompson's construction, draw a nondeterministic finite state machine equivalent to the following two regular expressions. **[2✓]**



5. Write a `bison` grammar for a simple language, described here. **[4✓]**
   (a) A program is a sequence of zero or more elements.
   (b) An element is an `ATOM` or a list.
   (c) A list is a left parenthesis followed by zero or more elements, followed by a right parenthesis.
   (d) The scanner returns only one of three kinds of tokens: `ATOM`, `'('`, `')'`. Do not code the scanner.

Use semantic actions to construct the entire program as a list, using the function `cons`: `$$=cons($1,$2)` will take an already constructed list (`$2`) and a new node (`$1`) and return the list with the new node prepended to the list. Thus, your rules must be right associative, so that for any list, the tail is constructed first. Do not use `adopt` from the project.

```
program : element program  { $$ = cons($1,$2); }
        |                  { $$ = nullptr; }
        ;
element : ATOM             { $$ = $1; }
        | '(' list         { $$ = $2; }
list    : element list     { $$ = cons($1,$2); }
        | ')'              { $$ = nullptr; }
```

6. Using the specifications for the code generation project, translate the following code into the intermediate language. ***Do not mangle names.*** **[6✓]**

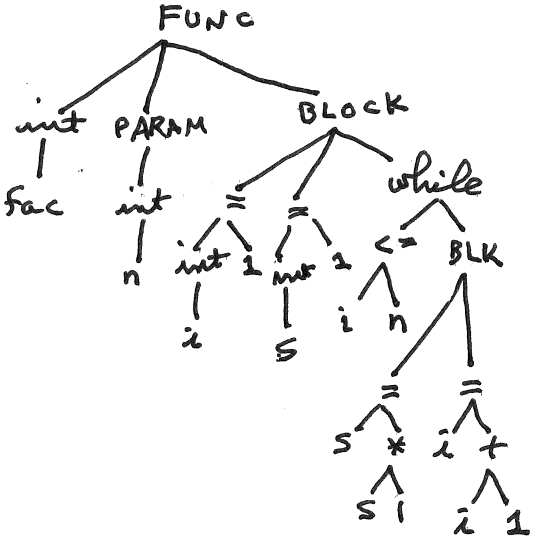
```
int fac (int n) {
   int i = 1;
   int s = 1;
   while (i <= n) {
      s = s * i;
      i = i + 1;
   }
}
```

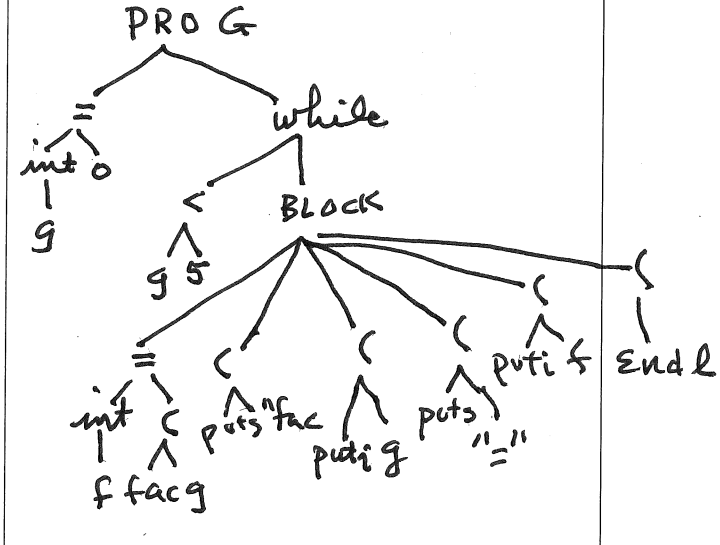
```
int fac(
    int n ){
    int i = 1;
    int s = 1;
while: t1 = i<=n;
    if(!t1) goto od
    t2 = s*i;
    s = t2;
    t3 = i+1;
    i = t3;
    goto while;
od:
    return;
}
```

```
int g = 0;
while (g < 5) {
   int f = fac (g);
   puts ("fac ");
   puti (g);
   puts (" = ");
   puti (f);
   endl();
}
```

```
int g;
void ocmain(){
    g = 0;
do: t1 = g <5;
    if(!t1) goto od;
    t2 = fac(g);
    int f = t2
    puts("fac")
    puti(g)
    puts(" = ")
    puti (f)
    endl()
    return
}
```

7. Draw abstract syntax trees for each of the above programs. Use the specifications for the parsing project. ***Points will be deducted for a messy diagram.*** **[4✓]**





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. What kind of automatic memory management does not work on a cyclic data structure ?
   (A) copying collection with semispaces
   (B) malloc and free
   (C) mark and sweep
   (D) reference counting

2. Which system call can be used by `malloc`(3) to extend the heap if it runs out of space ?
   (A) `brk`(2)
   (B) `fork`(2)
   (C) `lseek`(2)
   (D) `realloc`(3)

3. If a parser for ANSI C has the following two items on the stack and the lookahead symbol is **else**, then the symbol **else** should be [x] associative, and the precedence of the second rule should be the same as the **else**, so that we perform a [y] parsing action.
   $S \rightarrow$ **if** (E) S • **else** S
   $S \rightarrow$ **if** (E) S •
   (A) [x] = left, [y] = reduce
   (B) [x] = left, [y] = shift
   (C) [x] = right, [y] = reduce
   (D) [x] = right, [y] = shift

4. Byte-codes in class files generated by a Java compilation use what style ?
   (A) reverse Polish notation
   (B) three address code
   (C) two address code
   (D) x86-64 machine code

5. A C++ `unordered_set` will have an expected amortized lookup time of :
   (A) $O(1)$
   (B) $O(\log_2 n)$
   (C) $O(n)$
   (D) $O(n \log_2 n)$

6. What pattern can be described by a `bison` grammar, but not by a `flex` grammar ?
   (A) a sequence beginning and ending with a quote ("), with zero or more alphanumeric characters between the quotes.
   (B) a sequence of four or more decimal digits.
   (C) one or more `x`s followed by the same number of `y`s.
   (D) one `x` followed by zero or more `y`s.

7. What might possibly be the output from the following program ?
   ```
   void main(){printf("%p\n",main);}
   ```
   (A) `0x400796`
   (B) `Segmentation fault (core dumped)`
   (C) `UHå¿@`
   (D) `a.out: command not found`

8. The boundary tag method of memory allocation has what overhead per call to `malloc`(3) ?
   (A) 2 bytes
   (B) 2 pages
   (C) 2 pointers
   (D) 2 registers

9. If we represent a set of integers in the range 0 to 31 by a `uint32_t`, and `x` and `y` are two such sets, then the union of `x` and `y` can be coded as :
   (A) `x & y`
   (B) `x + y`
   (C) `x ^ y`
   (D) `x | y`

10. The intersection of the set of LALR(1) grammars with the set of ambiguous grammars is :
    (A) a non-empty subset of the set of context-free grammars.
    (B) identical to the set of LL(k) grammars.
    (C) the empty set.
    (D) the same as the set of context-free grammars.

11. An LALR(1) parser will perform how many shift operations when parsing a program with n tokens in the source code ?
    (A) $O(1)$
    (B) $O(\log_2 n)$
    (C) $O(n)$
    (D) $O(n \log_2 n)$

12. The following grammar is :
    $A \rightarrow A + A$
    $A \rightarrow A * A$
    $A \rightarrow i$
    (A) LL(1) but not LR(1).
    (B) LR(1) but not LL(1).
    (C) both LL(1) and LR(1).
    (D) neither LL(1) nor LR(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. Which of the following items was added to a CFSM state during a closure operation ?
   (A) $E \rightarrow \bullet\, E + T$
   (B) $E \rightarrow E \bullet + T$
   (C) $E \rightarrow E + \bullet\, T$
   (D) $E \rightarrow E + T \bullet$

2. Which of the following items will cause a reduction action to be added to a CFSM state which contains it ?
   (A) $E \rightarrow \bullet\, E + T$
   (B) $E \rightarrow E \bullet + T$
   (C) $E \rightarrow E + \bullet\, T$
   (D) $E \rightarrow E + T \bullet$

3. which of the following items will cause a shift transition to be added to a CFSM state which contains it ?
   (A) $E \rightarrow \bullet\, E + T$
   (B) $E \rightarrow E \bullet + T$
   (C) $E \rightarrow E + \bullet\, T$
   (D) $E \rightarrow E + T \bullet$

4. In a language with nested procedures, the inner procedure can find the local stack frame of the outer procedure by using :
   (A) argument pointer
   (B) dynamic link
   (C) return address
   (D) static link

5. After an x86-64 function prolog of :
   ```
   pushq   %rbp
   movq    %rsp, %rbp
   ```
   Where is the function's return address ?
   (A) `-8(%rbp)`
   (B) `-8(%rsp)`
   (C) `8(%rbp)`
   (D) `8(%rsp)`

6. The x86-64 has x integer registers, each of which hold y bits.
   (A) $x = 8$ and $y = 16$
   (B) $x = 8$ and $y = 32$
   (C) $x = 16$ and $y = 32$
   (D) $x = 16$ and $y = 64$

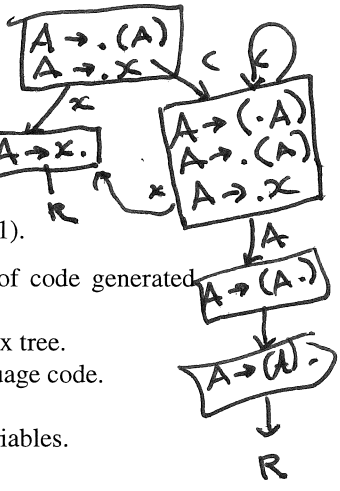
7. How many tokens are there in the following C code ?
   ```
   printf ("Hello, world.\n"); /* Say hello. */
   ```
   (A) 3
   (B) 5
   (C) 7
   (D) 9

8. Given a grammar $G = \langle V_N, V_T, P, S \rangle$, if each state of the LALR(1) parse table is a row, how many columns are there ?
   (A) $|V_N|$
   (B) $|V_T|$
   (C) $|V_N \cup V_T|$
   (D) $|V_N \cap V_T|$

9. In doing local register allocation, which category of registers is the cheapest to allocate ?
   (A) dead, no save
   (B) dead, save
   (C) live, no save
   (D) live, save

10. Dominators of a flow graph are used to :
    (A) detect tail calls.
    (B) find all induction variables.
    (C) identify natural loops.
    (D) improve efficiency of switch statements.

11. The following grammar :
    $A \rightarrow (A)$
    $A \rightarrow x$
    (A) is LR(0) but not SLR(1).
    (B) is SLR(1) but not LR(0).
    (C) is both LR(0) and SLR(1).
    (D) is neither LR(0) nor SLR(1).



12. The most important function of code generated by `flex` is :
    (A) building the abstract syntax tree.
    (B) generating assembly language code.
    (C) identifying lexemes.
    (D) identifying undeclared variables.