Indian Institute of Information Technology, Kalyani
Mid-Semester Examination 2019

Subject: Compiler Design — Time: 1 hr 30 min.
Paper Code: CS-501 — Full Marks: 45

Instructions : **There are four (4) questions. Answer all of them.**

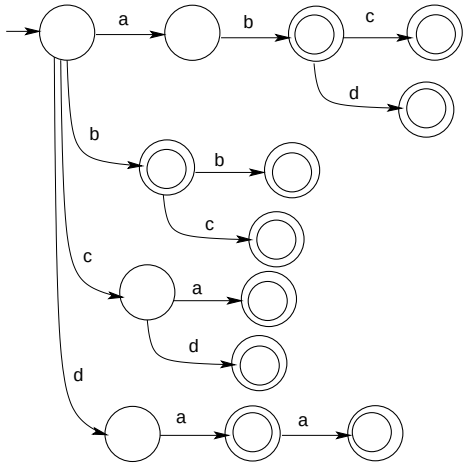
1. Answer each question with a brief explanation. **[7 × 3]**

   (a) Consider the set of tokens, {`ab, abc, abd, b, bc, bb, ca, cd, da, daa`}. What is the sequence of maximal length tokens generated from the input "`abcabbdaadacaabcbb`".

   Ans. {`abc ab b daa da ca abc bb`}.

   (b) Draw a deterministic transition diagram for the set of tokens in (1a). Clearly mark the *start state* and the *final states*.

   Ans.



   (c) Give the production rules of an *ambiguous* CFG for expressions over $\Sigma = \{id, fc, +, *, =,), (\}$, where *id* is an *identifier* and *fc* is a *float constant*. Operators '=' (assignment), '+', '$*$', and parenthesis ')', '(' have their usual meaning and purpose. You can use only one non-terminal E.

   Ans.

$$E \rightarrow id = E \mid E + E \mid E * E \mid (E) \mid id \mid fc$$

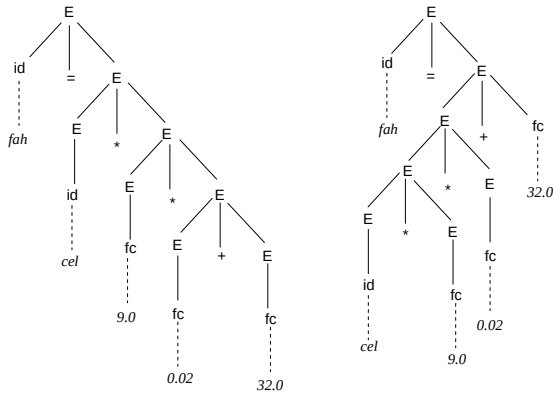
(d) Give an *unambiguous* CFG *equivalent* to the CFG of (1c). The operator-precedence is '$= < + < *$'. Both '$+$' and '$*$' are *left-associative*, but '$=$' is *right-associative*. Parenthesis ')', '(' are used to overwrite precedence and associativity.

Ans.

$$\begin{aligned} E &\rightarrow id = E \mid P \\ P &\rightarrow P + M \mid M \\ M &\rightarrow M * B \mid B \\ B &\rightarrow (E) \mid id \mid fc \end{aligned}$$

(e) Draw two *parse trees* corresponding to the string "`fah = cel * 9.0 * 0.02 + 32.0`" in the grammar of (1c).

Ans.



(f) Remove *left recursion* from the CFG $G_1 = (\{a, b, c, d\}, \{A, B\}, P, A)$, where $P = \{A \rightarrow Aa \mid Aab \mid Bc,\ B \rightarrow BAa \mid d\}$.

Ans. $A \rightarrow BcA',\ A' \rightarrow aA' \mid abA' \mid \varepsilon,\ B \rightarrow dB',\ B' \rightarrow AaB' \mid \varepsilon.$

(g) Consider the following state-transition table of a DFA over an alphabet $\{0, 1, 2\}$. The states are $\{q_\varepsilon, q_0, q_1, q_2, q_3\}$.

CS	NS		
	0	1	2
q_ε	0	1	2
q_0	0	1	2
q_1	3	0	1
q_2	2	3	0
q_3	1	2	3

If each table entry takes *one byte*, show that the table can be compressed to a 1D-vector of size *six bytes*. Show the displacement for each row (state). (*CS: current state*, *NS: next state*)

Ans. The state-transition vector is

0	1	2	3	0	1

The displacement in the state-transition vector for each state is

q_ε	q_0	q_1	q_2	q_3
0	0	3	2	1

2. Extract a C program from the following x86-64 assembly code where the memory locations of variables `a, b` and `c` are `Memory[rbp - 20]`, `Memory[rbp - 16]` and `Memory[rbp - 12]` respectively. Assume that the variable 'a' contains a positive integer n. What is finally value computed in the variable 'b'? **[6]**

```
  movl  $1, -12(%rbp)
  movl  $0, -16(%rbp)
  jmp  .L2
.L3:
  movl  -12(%rbp), %eax
  addl  %eax, -16(%rbp)
  addl  $2, -12(%rbp)
.L2:
  movl  -20(%rbp), %eax
  cmpl  %eax, -12(%rbp)
  jle  .L3
```

Ans.

```
  movl  $1, -12(%rbp)     # c = 1
  movl  $0, -16(%rbp)     # b = 0
  jmp  .L2                # goto .L2
.L3:
  movl  -12(%rbp), %eax   # eax = Mem[rbp-12] (c)
  addl  %eax, -16(%rbp)   # Mem[rbp-16](b) = b + eax (c)
  addl  $2, -12(%rbp)     # Mem[rbp-2](c) = c + 2
.L2:
  movl  -20(%rbp), %eax   # eax = Mem[rbp-20](a)
  cmpl  %eax, -12(%rbp)   # compare Mem[rbp-12](c), eax (a)
  jle  .L3                # if c <= a goto .L3
```

The C code is

```
c=1;
b=0;
L:
if(c <= a) {
   b = b + c;
   c = c + 2;
   goto L;
}
```

Equivalent C code

```
c=1;
b=0;
while(c <= a) {
   b = b + c;
   c = c + 2;
}
```

The sum of odd integers in the range of 1 to n is computed in `b`.

3. A relocatable ELF file is mapped to the address space of a process.

   (a) The variable `elfhP` of type `Elf64_Ehdr *` stores the starting address of the map. How do you use the following fields of the ELF header (`Elf64_Ehdr`) to find the address of the (i) *section header table* and (ii) the address of its string table entry.
   `e_shoff, e_shstrndx, e_shentsize`.

   (b) The variable `shP` of type '`Elf64_Shdr *`' stores the address of the section header table and `textOff` is the offset of "`.text`" within the section header string table. How do you find the address of the section header corresponding to `.text` section using `sh_name` field of the section header structure? **[3+3]**

   Ans.

   (a) The address of *section header table* is `elfP -> e_shoff`. The address of the section header entry of its string table is
   `elfhP->e_shoff+elfhP->e_shstrndx*elfhP->e_shentsize`.

   (b) The section header entry for the `.text` section can be obtained by the C code
   `while(shP->sh_name != textOff) shP = shP+1;`

4. Consider the regular expression $0^*1(10^*1 + 01^*0)^*$.

   (a) Construct a DFA corresponding to the regular expression whose states are sets of *dotted regular expressions (items)*. The start state is $\mathbf{q}_0 : \{(\bullet 0)^*1(10^*1 + 01^*0)^*,\ 0^* \bullet 1(10^*1 + 01^*0)^*\}$.

   (b) Construct the syntax tree corresponding to the augmented regular expression: $(0^*1(10^*1 + 01^*0)^*)\#$. Decorate each node with *firstpos* and *lastpos* data. Compute *followpos* for different positions and draw the corresponding *non-deterministic finite automaton (NFA)*.

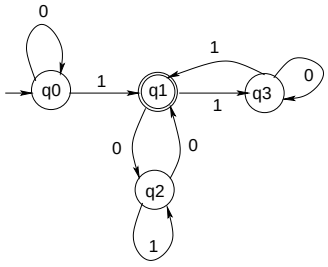
[5+7]

Ans.

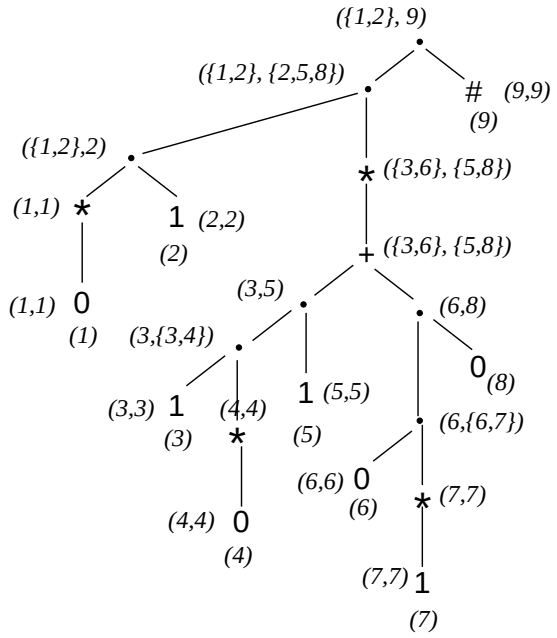
(a) The states and the transition table are:

q_0 : $\{(\bullet 0)^*1(10^*1 + 01^*0)^*,$
$0^* \bullet 1(10^*1 + 01^*0)^*\}$
q_1 : $\{0^*1(\bullet 10^*1 + 01^*0)^*$
$0^*1(10^*1 + \bullet 01^*0)^*$
$0^*1(10^*1 + 01^*0)^* \bullet\}$
q_2 : $\{0^*1(10^*1 + 0(\bullet 1)^*0)^*$
$0^*1(10^*1 + 01^* \bullet 0)^*\}$
q_3 : $\{0^*1(1(\bullet 0)^*1 + 01^*0)^*$
$0^*1(10^* \bullet 1 + 01^*0)^*\}$

CS	NS on Input	
	0	1
$\rightarrow q_0$	q_0	q_1
$* q_1$	q_2	q_3
q_2	q_1	q_2
q_3	q_3	q_1



(b) The abstract syntax tree is as follows:



followpos() for different positions are as follows:

Position	Follow Positions
1	$\{1, 2\}$
2	$\{3, 6, 9\}$
3	$\{4, 5\}$
4	$\{4, 5\}$
5	$\{3, 6, 9\}$
6	$\{7, 8\}$
7	$\{7, 8\}$
8	$\{3, 6, 9\}$
9