|  |  |
| --- | --- |
| No. of Pages | 5 |
| No. of Questions | 9 |

**Student Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Department of Computer Science and Engineering**

**Final Examination Spring 2016**

**CSE 420: Compiler Design**

**Total Marks: 70 Time Allowed: 3.00 Hour**

|  |
| --- |
| * You HAVE TO RETURN this question paper and the answer script at the end of the exam. Your script will not be checked unless you do so. * You are not allowed to communicate with any other candidate in any way what so ever. |

**Section 01 (There are 4 questions, answer any 3 out of them) [12 \*3 = 36]**

1. **a.** Draw the block diagram of basic compilation phases. [3]

b. Discuss the comparative advantages and disadvantages of following the intermediate code representations: [3]

**i)** Quadruples

**ii)** Triples

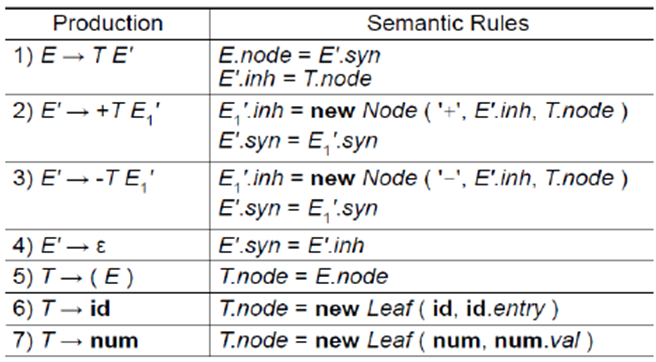
**iii)** Indirect triples

**c.** Determine the target statements of following code segment using the techniques of back-patching: [6]

|  |  |
| --- | --- |
| if(m>2 && m<50 && m%3==0){  a=a+2;  b++;  m/=3; }  else{ a=a+5;  b--; } |  |

**2. a.** Define variable and register descriptor? [2]

**b.** Consider the following SDD: [4+2=8]



**i)** Draw the annotated parse tree for the expression: *3+ (5-7) +4*.

**ii)** Draw the dependency graph.

**d.** Draw DFA for the regular expression: {00}\*{11}\*. [2]

**3. a.** Consider following grammar: [3+3+3=9]

*S-> N*

*N->NB| B*

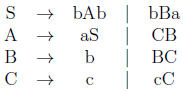
*B->1|0*

**i)** Construct the LR(0) Automation of the grammar above.

**ii)** Construct the parse table.

**iii)** Parse the string 110110 showing all the necessary steps.

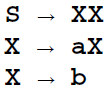
**b.** Consider the following grammar: [2]



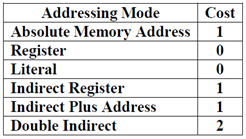
Give two reasons why this grammar is not LL (1).

**c.** Define symbol table. [1]

**4. a.** With LALR (look ahead LR) parsing, we can reduce the number of states in an LR(1) parser. Justify the above statement using the following grammar: [5]



**b.** Translate the expression *(a[i]+b)\*c* into Quadruples. Also calculate the total cost of generated code using following cost model (try to scale down the cost as much as possible). [2+3]



**c.** What is the problem with a production with an immediate left-recursion in a grammar? How can we eliminate left recursion? [2]

**Section 02: (There are 3 questions, answer any 2 out of them) [10\*2 =20]**

**1. a.** Convert following DFA to regular expression, showing all the steps: [3]

**b.** Draw the control flow block diagram for following statements: [5]

|  |  |
| --- | --- |
| while(x>0 && x<50){  if(a%2==0)  a= a+x;  else  a= a\*x;  if(a>400)  break;  } |  |

**c.** Write short notes on the following: [1+1]

**i)** Activation Tree

**ii)** Handle Pruning

**2. a.** “Every s-attributed definition is L-attributed” – Justify your answer. [2]

**b.** Using Thompson’s construction draw the NFA for the following regular expression: (ab)\*(c|d). [4]

**c.** Draw a block diagram showing the caller-callee responsibilities in the construction of activation record. [4]

**3. a.** What would be the transition diagram (TD) for email address of the form (*abcde.1999.cse@bracu.ac.bd*), look like? [2]

**b.** Consider the grammar with the set of terminals: [2+3]

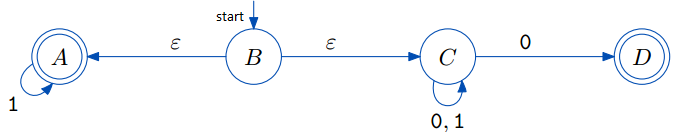
*E → E + T | E – T | T*

*T → T \* F | T / F | F*

*F → (E) | id*

Remove left-recursion from the grammar and find the First and Follow sets for each non-terminal of the modified grammar.

**c.** Convert the following NFA to DFA using subset construction method. [3]



**Section 03: (There are 2 questions, answer any 1 out of them) [1\*14 = 14]**

**1.** **a.** Define following terms with appropriate examples: [2]

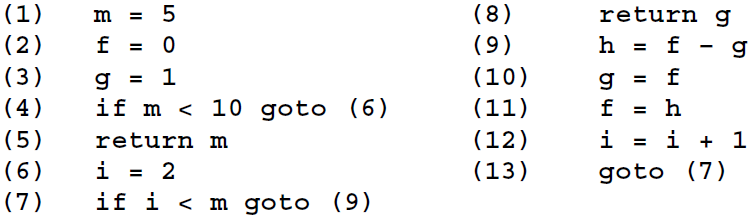
**i)** Common sub-expression elimination,

**ii)** Dead code elimination.

**b.** Find the LL(1) parsing table for the following grammar and also parse the string “*acbdbb”*: [6]



**c.** Consider the code segment below: [3+3]



1. Construct the control flow graph.
2. Define Use-set, define-set of each individual block.

**1. a.** Convert RE *1(00+01)\*0* to corresponding DFA (using first-pos, last-pos and follow-pos). [9]

**b.** With the help of following example describe the “next-use” algorithm. [3]



**c.** Eliminate the common sub-expressions from following control flow graph: [2]

