Graphic Era HILL UNIVERSITY
Established by an Act of the State Legislature of Uttarakhand (Adhiniyam Sankhya 12 of 2011)
University under section 2(f) of UGC Act, 1956

# End Term (Even) Semester Examination May-June 2025

Roll no. 2294038

Name of the Program and semester: B.Tech.CSE (Core/AI-ML)
Name of the Course: Compiler Design
Course Code: TCS-601
Time: 3 hour

Maximum Marks: 100

**Note:**

*(i)* All the questions are compulsory.
*(ii)* Answer any two sub questions from a, b and c in each main question.
*(iii)* Total marks for each question is 20 (twenty).
*(iv)* Each sub-question carries 10 marks.

Q1. (2X10=20 Marks)

a. What are the phases of compilers? Categorize them into front end and back end of the compiler. Discuss the action taken by every phase of the compiler on the following string: (CO1)

A = B*C+D/E

b. Discuss the working of Lex compiler. Also, explain the format of its input specification file. (CO5)

c. Explain the need for input buffering in lexical analysis. Consider a source program stored in a file. Design a simulation (in pseudocode) of the input buffer handling using the two-buffer scheme. Clearly explain how buffer switching and token formation are handled. (CO4)

Q2. (2X10=20 Marks)

a. Consider the following context-free grammar. (CO2)

$S \rightarrow ( )$
$S \rightarrow a$
$S \rightarrow ( A )$
$A \rightarrow S$
$A \rightarrow A , S$

(i) Show precisely why this grammar is not LL(1).
(ii) Rewrite this grammar to make it suitable for predictive parsing.
(iii) On the basis of your revised grammar from part (ii), write a recursive procedure S that parses this grammar by recursive descent. Your procedure may be written in C, C++, Java or pseudocode. You may assume the existence of a global variable token that holds the next input token, a function advance() that reads the next token into token, and a function error that may be called if the input is not in the language generated by the grammar.

b. Consider the following grammar: (CO2, CO4)

**S→(L)/a**
**L→L,S/S**

For this grammar, design the CLR(1) parsing table. Also describe the working of automatic parser Generator.

c. Consider the following grammar (CO2)

S→ Aab , A→cA / cb

Apply Lead and Last method to design operator precedence table and parse the string "ccbab" using operator precedence parsing technique.

Q3. (2X10=20 Marks)

Graphic Era HILL UNIVERSITY
Established by an Act of the State Legislature of Uttarakhand (Adhiniyam Sankhya 12 of 2011)
University under section 2(f) of UGC Act, 1956

## End Term (Even) Semester Examination May-June 2025

a. What do you mean by backpatching? Apply backpatching to write down the SDT for the following Boolean expressions and Flow-of-Control statements. (CO3)

E→E1 or E2
S→if E then S1 else S2
S→ while E do S1

b. Translate the following code fragment into 3AC and implement the generated 3AC using Quadruples and Triples. (CO1, CO3)

```
switch(a+b)
  {
      case 2: {x=y; break; }
      case 5: {switch x
                {
                  case 0: { a= b+1; break; }
                  case 1: {a= b+3; break; }
                  default: {a=2; break; }
                }

                break;
              }
      case 9: {x= y-1; break; }
      default: {a=2; break; }
    }
```

c. Consider the following SDD over the grammar defined by G = ({S, A, Sign}, S, {',', '-', '+', 'n'}, P) with P the set of production and the corresponding semantic rules depicted below. There is a special terminal symbol "n" that is lexically matched by any string of one numeric digit and whose value is the numeric value of its decimal representation. For the non-terminal symbols in G we have defined two attributes, sign and value. The non-terminal A has these two attributes whereas S only has the value attribute and Sign only has the sign attribute. (CO3)

| | | |
|---|---|---|
| **S → A Sign** | ‖ | **S.val = A.val; A.sign = Sign.sign; print(A.val);** |
| **Sign → +** | ‖ | **Sign.sign = 1** |
| **Sign → -** | ‖ | **Sign.sign = 0** |
| **A → n** | ‖ | **A.val = value(n)** |
| **A → $A_1$ , n** | ‖ | **$A_1$.sign = A.sign;**<br>**if(A.sign = 1) then**<br>**A.val = min ($A_1$.val,value(n));**<br>**else**<br>**A.val = max($A_1$.val,value(n));** |

For this SDD answer to the following questions:

i) Explain the overall operation of this syntax-directed definition and indicate (with a brief explanation) which of the attributes are either synthesized or inherited.

ii) Give an attributed parse tree for the source string "5,2,3-" and evaluate the attributes in the attributed parse tree depicting the order in which the attributes need to be evaluated (if more than one order is possible indicate one.)

iii) Suggest a modified grammar and actions exclusively using synthesized attributes. Explain its basic operation.

Graphic Era HILL UNIVERSITY
Established by an Act of the State Legislature of Uttarakhand (Adhiniyam Sankhya 12 of 2011)
University under section 2(f) of UGC Act, 1956

## End Term (Even) Semester Examination May-June 2025

Q4. (2X10=20 Marks)

a. What are the various storage allocation techniques available? What are their importance in compiler design? (CO3)

b. Write short notes on the following: (CO1, CO3)

i. Symbol table data structure
ii. Activation record
iii. Activation tree
iv. Dangling references

c. How can a sequence of 3AC statements be transformed into basic block? Design DAG for the following code segment: (CO1, CO3)

```
(1)   t1:=4*i
(2)   t2:=a[t1]
(3)   t3:=4*i
(4)   t4:=b[t3]
(5)   t5:=t2*t4
(6)   t6:=prod+t5
(7)   prod:=t6
(8)   t7:=i+1
(9)   i:=t7
(10)  if i<=10 goto (3)
```

Q5. (2X10=20 Marks)

a. Apply gencode() function to design the machine code for the following program fragment, where a and b are array of size 20 x 20 having low1=low2=1, and there are four bytes per word. (CO1, CO6)

```
begin
    prod = 0;
      x=1;
      y=1;
    do
        begin
          prod= prod + a[x,y] * b[x,y]
           x= x+1;
           y= y+1;
         end
  while x<= 20 and y<= 20;
end
```

b. Compare and contrast the different approaches to compiler development, such as monolithic, modular, and incremental approaches. How does the choice of development approach influence the selection and use of tools in the compiler development environment? Support your answer with examples. (CO6)

c. Perform the optimization on the following code fragment. (CO1, CO6)

```
void quicksort (m,n)
    int m,n;
    {
      int i,j;
```

```
        int v,x;
      if (n <= m) return;
/* fragment begins here */
              i = m-1;  j = n;   v = a[n];
      while (1) {
       do i = i+1; while (a[i] < v);
       do j = j-1; while (a[j] > v);
        if (i >= j) break;
        x = a[i];   a[i] = a[j];  a[j] = x;
              }
      x = a[i];   a[i] = a[n];   a[n] = x;
 /* fragment ends here */
     quicksort (m,j) ;  quicksort (i+1,n);
         }
```