**(1)**

**(A)1.** Write a program to generate Symbol table of a two-pass Assembler for the given Assembly language source code.

|  |
| --- |
| **INPUT/CODE**  START 100  READ A  READ B  LOOP MOVER AREG, A  MOVER BREG, B  COMP BREG, =’2’  BC GT, LOOP  BACK SUB AREG, B  COMP AREG, =’5’  BC LT, BACK  STOP  A DS 1  B DS 1  END |

**++++++++++++++++++++++++++++++++++++**

**(2)**

**(B)1.**Write a program to generate Symbol table of a two-pass Assembler for the given Assembly language source code.

|  |
| --- |
| **INPUT/CODE**  START 150  READ D  READ E  LOOP MOVER AREG, D  MOVER BREG, E  COMP BREG, =’20’  BC GT, LOOP  BACK SUB AREG, E  COMP AREG, =’50’  BC LT, BACK  STOP  D DS 1  E DS 1  END |

**++++++++++++++++++++++++++++++++++++**

**(3)**

**(C) 1.** Write a program to generate Symbol table of a two-pass Assembler for the given Assembly language source code.

|  |
| --- |
| **INPUT/CODE**  START 180  READ M  READ N  LOOP MOVER AREG, M  MOVER BREG, N  COMP BREG, =’200’  BC GT, LOOP  BACK SUB AREG, M  COMP AREG, =’500’  BC LT, BACK  STOP  M DS 1  N DS 1  END |

**++++++++++++++++++++++++++++++++++++**

**(4)**

**(A)2.** Write a program to generate Literal table of a two-pass Assembler for the given Assembly language source code.

|  |
| --- |
| **INPUT/CODE**  START 100  READ A  READ B  MOVER AREG, =’50'  MOVER BREG, =’60’  ADD AREG, BREG  LOOP MOVER CREG, A  ADD CREG, ='10'  COMP CREG, B  BC LT, LOOP  NEXT SUB AREG, ='10'  COMP AREG, B  BC GT, NEXT  STOP  A DS 1  B DS 1  END |

**++++++++++++++++++++++++++++++++++++**

**(5)**

**(B)2.** Write a program to generate Literal table of a two-pass Assembler for the given Assembly language source code.

|  |
| --- |
| START 200  READ X  READ Y  MOVER AREG, =’5'  MOVER BREG, =’6’  ADD AREG, BREG  LOOP MOVER CREG, X  ADD CREG, ='1'  COMP CREG, Y  BC LT, LOOP  NEXT SUB AREG, ='1'  COMP AREG, Y  BC GT, NEXT  STOP  X DS 1  Y DS 1  END |

**++++++++++++++++++++++++++++++++++++**

**(6)**

**(C)2.** Write a program to generate Literal table of a two-pass Assembler for the given Assembly language source code.

|  |
| --- |
| START 300  READ M  READ N  MOVER AREG, =’51'  MOVER BREG, =’61’  ADD AREG, BREG  LOOP MOVER CREG, M  ADD CREG, ='11'  COMP CREG, N  BC LT, LOOP  NEXT SUB AREG, ='11'  COMP AREG, N  BC GT, NEXT  STOP  M DS 1  N DS 1  END |

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**(7)**

**(A)3**.Write a program to generate Pool table of a two-pass Assembler for the given Assembly language source code.

|  |
| --- |
| **INPUT/CODE**  START 100  READ A  MOVER AREG, ='1'  MOVEM AREG, B  MOVER BREG, ='6'  ADD AREG, BREG  COMP AREG, A  BC GT, LAST  LTORG  NEXT SUB AREG, ='1'  MOVER CREG, B  ADD CREG, ='8'  MOVEM CREG, B  PRINT B  LAST STOP  A DS 1  B DS 1  END |

**++++++++++++++++++++++++++++++++++++**

**(8)**

**(B)3**.Write a program to generate Pool table of a two-pass Assembler for the given Assembly language source code.

|  |
| --- |
| **INPUT/CODE**  START 200  READ X  MOVER AREG, ='10'  MOVEM AREG, Y  MOVER BREG, ='60'  ADD AREG, BREG  COMP AREG, X  BC GT, LAST  LTORG  NEXT SUB AREG, ='10'  MOVER CREG, Y  ADD CREG, ='80'  MOVEM CREG, Y  PRINT B  LAST STOP  X DS 1  Y DS 1  END |

**++++++++++++++++++++++++++++++++++++**

**(9)**

**(C)3**.Write a program to generate Pool table of a two-pass Assembler for the given Assembly language source code.

|  |
| --- |
| **INPUT/CODE**  START 300  READ M  MOVER AREG, ='11'  MOVEM AREG, N  MOVER BREG, ='61'  ADD AREG, BREG  COMP AREG, M  BC GT, LAST  LTORG  NEXT SUB AREG, ='11'  MOVER CREG, N  ADD CREG, ='81'  MOVEM CREG, N  PRINT Y  LAST STOP  M DS 1  N DS 1  END |

**++++++++++++++++++++++++++++++++++++**

**(10)**

**(A)4.**Write a program to generate Intermediate code of a two-pass Assembler for the given Assembly language source code.

|  |
| --- |
| **INPUT/CODE**  START 100  READ A  READ B  MOVER AREG, A  SUB AREG, B  STOP  A DS 1  B DS 1  END |

**++++++++++++++++++++++++++++++++++++**

**(11)**

**(B)4.**Write a program to generate Intermediate code of a two-pass Assembler for the given Assembly language source code.

|  |
| --- |
| **INPUT/CODE**  START 200  READ M  READ N  MOVER AREG, M  SUB AREG, N  STOP  M DS 1  N DS 1  END |

**++++++++++++++++++++++++++++++++++++**

**(12)**

**(C)4.**Write a program to generate Intermediate code of a two-pass Assembler for the given Assembly language source code.

|  |
| --- |
| **INPUT/CODE**  START 300  READ M  READ N  MOVER AREG, M  SUB AREG, N  STOP  M DS 1  N DS 1  END |

**++++++++++++++++++++++++++++++++++++**

**(13)**

**(A)5.**Write a program to generate Intermediate code of a two-pass Macro processor.

|  |  |
| --- | --- |
| **INPUT/CODE**  LOAD A  MACRO ABC  LOAD p  SUB q  MEND  STORE B  MULT D  MACRO ADD1 ARG  LOAD X  STORE ARG  MEND  …continued… | …continued…  LOAD B  MACRO ADD5 A1, A2, A3  STORE A2  ADD1 5  ADD1 10  LOAD A1  LOAD A3  MEND  ADD1 t  ABC  ADD5 D1, D2, D3  END |

**++++++++++++++++++++++++++++++++++++**

**(14)**

**(B)5.**Write a program to generate Intermediate code of a two-pass Macro processor.

|  |  |
| --- | --- |
| **INPUT/CODE**  STORE P  LOAD Q  MACRO PCG  LOAD m  ADD n  MEND  LOAD H  LOAD K  MACRO ADDi PAR  LOAD A  STORE PAR  MEND  …continued… | …continued…  DIV R  MACRO ADDii V1, V2, V3  STORE V2  ADDi 12  ADDi 7  LOAD V1  LOAD V3  MEND  PCG  ADDii Q1, Q2, Q3  ADDi w  END |

**++++++++++++++++++++++++++++++++++++**

**(15)**

**(C)5.**Write a program to generate Intermediate code of a two-pass Macro processor.

|  |  |
| --- | --- |
| **INPUT/CODE**  LOAD F  STORE E  MACRO SRS  LOAD s  SUB t  MEND  STORE k  MACRO ADD3 XYZ  LOAD U  STORE XYZ  MEND  …continued… | …continued…  Add m  MACRO ADD1 Si, Sii, Siii  LOAD Sii  ADD3 1  ADD3 11  STORE Si  STORE Siii  MEND  SRS  ADD1 C1, C2, C3  ADD3 q  END |

**++++++++++++++++++++++++++++++++++++**

**(16)**

**(D)5.**Write a program to generateIntermediate code of a two-pass Macro processor.

|  |  |
| --- | --- |
| **INPUT/CODE**  LOAD J  STORE M  MACRO EST  LOAD e  ADD d  MEND  LOAD S  MACRO SUB4 ABC  LOAD U  STORE ABC  MEND | …continued….  LOAD P  ADD V  MACRO ADD7 P4, P5, P6  LOAD P5  SUB4 XYZ  SUB 8  SUB 2  STORE P4  STORE P6  MEND  EST  ADD7 C4, C5, C6  SUB4 z  END |

**++++++++++++++++++++++++++++++++++++**

**(17)**

**(A)6.**Write a program to generate MDT(Macro Definition Table) of a two-pass Macro processor.

|  |  |
| --- | --- |
| **INPUT/CODE**  LOAD A  STORE B  MACRO ABC  LOAD p  SUB q  MEND  MACRO ADD1 ARG  LOAD X  STORE ARG  MEND  ….Continued…. | …continued….  MACRO ADD5 A1, A2, A3  STORE A2  ADD1 5  ADD1 10  LOAD A1  LOAD A3  MEND  ABC  ADD5 D1, D2, D3  END |

**++++++++++++++++++++++++++++++++++++**

**(18)**

**(B)6.**Write a program to generate MDT(Macro Definition Table) of a two-pass Macro processor.

|  |  |
| --- | --- |
| **INPUT/CODE**  STORE P  LOAD Q  MACRO PCG  LOAD m  ADD n  MEND  MOV S  MACRO ADDi PAR  LOAD A  STORE PAR  MEND  …continued… | …continued…  DIV B  MACRO ADDii V1, V2, V3  STORE V2  ADDi 12  ADDi 7  LOAD V1  LOAD V3  MEND  PCG  ADDii Q1, Q2, Q3  END |

**++++++++++++++++++++++++++++++++++++**

**(19)**

**(A)7.**Write a program to generate MNT(Macro Name Table) of a two-pass Macro processor.

|  |  |
| --- | --- |
| **INPUT/CODE**  LOAD F  STORE E  MACRO SRS  LOAD s  SUB t  MEND  MACRO SRS XYZ  LOAD U  STORE XYZ  MEND | MACRO ADD1 Si, Sii, Siii  LOAD Sii  ADD3 1  SRS 11  STORE Si  STORE Siii  MEND  SRS  ADD1 C1, C2, C3  END |

**++++++++++++++++++++++++++++++++++++**

**(20)**

**(B)7.**Write a program to generate MNT(Macro Name Table) of a two-pass Macro processor.

|  |  |
| --- | --- |
| **INPUT/CODE**  LOAD J  STORE M  MACRO EST1  LOAD e  ADD d  MEND  MACRO EST ABC  EST1  STORE ABC  MEND | MACRO ADD7 P4, P5, P6  LOAD P5  EST 8  SUB4 2  STORE P4  STORE P6  MEND  EST  ADD7 C4, C5, C6  END |

**++++++++++++++++++++++++++++++++++++**

**(21)**

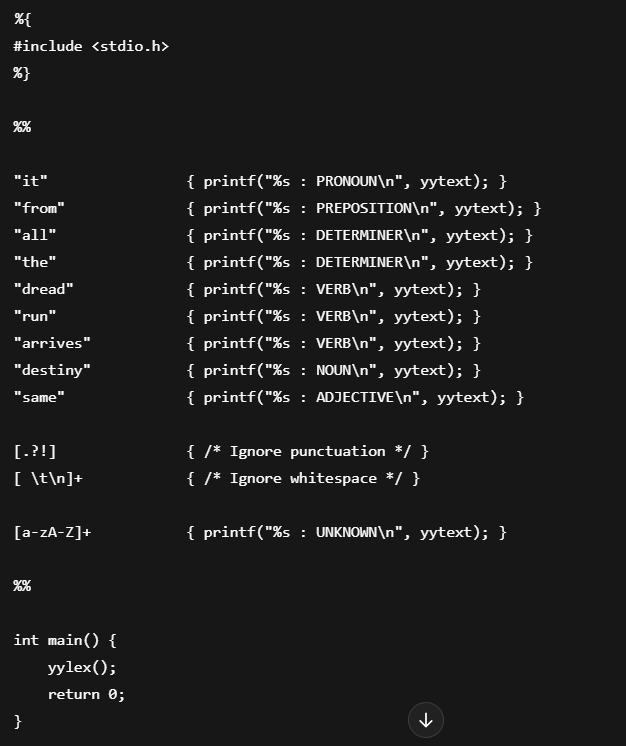
**(A)8.**Write a program using LEX Tool, to implement a lexical analyzer for parts of speech for given English language without Symbol table.

**INPUT**

Dread it. Run from it.

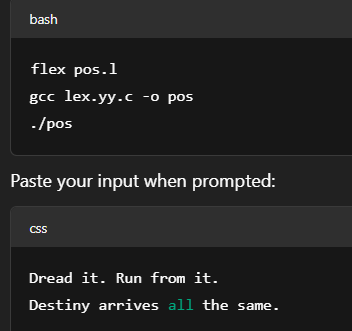
Destiny arrives all the same.

🡪 ✅ **LEX Code: POS Tagger (No Symbol Table)**



**📌 How to Run**

1. Save this code to a file named pos.l
2. Use Flex to compile:



**++++++++++++++++++++++++++++++++++++**

**(22)**

**(B)8.**Write a program using LEX Tool, to implement a lexical analyzer for parts of speech for given English language without Symbol table.

**INPUT**

Hello! How are you?

I’m fine, Thank You.

**++++++++++++++++++++++++++++++++++++**

**(23)**

**(C)8.**Write a program using LEX Tool, to implement a lexical analyzer for parts of speech for given English language without Symbol table.

**INPUT**

The important thing is to not stop questioning,

Curiosity has its own reason for existing.

**++++++++++++++++++++++++++++++++++++**

**(24)**

**(D)8.**Write a program using LEX Tool, to implement a lexical analyzer for parts of speech for given English language without Symbol table.

**INPUT**

If you never light the cigarette,

you never give the thing that can kill you the power it needs to kill you.

**++++++++++++++++++++++++++++++++++++**

**(25)**

**(E)8.**Write a program using LEX Tool, to implement a lexical analyzer for parts of speech for given English language without Symbol table.

**INPUT**

Every generation imagines itself to be more intelligent than the one that went before it, and wiser than the one that comes after it.



**++++++++++++++++++++++++++++++++++++**

**(26)**

**(F)8.**Write a program using LEX Tool, to implement a lexical analyzer for parts of speech for given English language without Symbol table.

**INPUT**

Human Beings have a strong dramatic instinct toward binary thinking, a basic urge to divide things into two distinct groups, with nothing but an empty gap in between.

**++++++++++++++++++++++++++++++++++++**

**(27)**

**(A)9.**Write a program using LEX Tool, to implement a lexical analyzer for given C programming language without Symbol table.

**INPUT**

{

int m=10,n=2,o;

o = m – n;

}

**++++++++++++++++++++++++++++++++++++**

**(28)**

**(B)9.**Write a program using LEX Tool, to implement a lexical analyzer for given C programming language without Symbol table.

**INPUT**

{

char = a;

printf(“Hello World”);

}

**++++++++++++++++++++++++++++++++++++**

**(29)**

**(C)9.**Write a program using LEX Tool, to implement a lexical analyzer for given C programming language without Symbol table.

**INPUT**

{

int d=10,t=2;

int s;

s=d/t;

}

**++++++++++++++++++++++++++++++++++++**

**(30)**

**(D)9.**Write a program using LEX Tool, to implement a lexical analyzer for given C programming language without Symbol table.

**INPUT**

{

int a=3;

int b=4;

float c;

c = (a\*a + b\*b) \*2

}

**++++++++++++++++++++++++++++++++++++**

**(31)**

**(E)9.** Write a program using LEX Tool, to implement a lexical analyzer for given C programming language without Symbol table.

**INPUT**

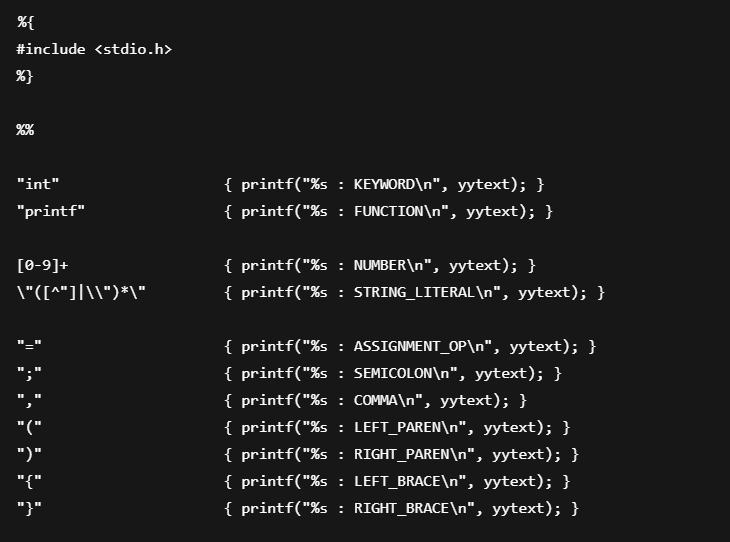
{

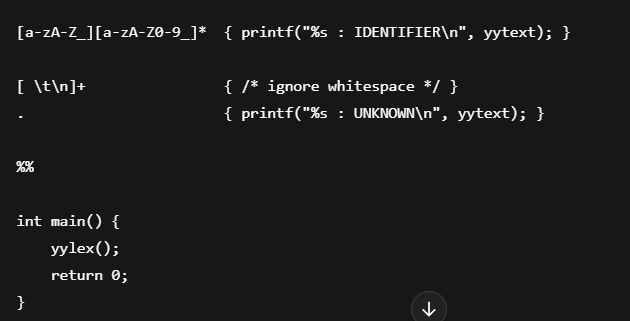
int total =100;

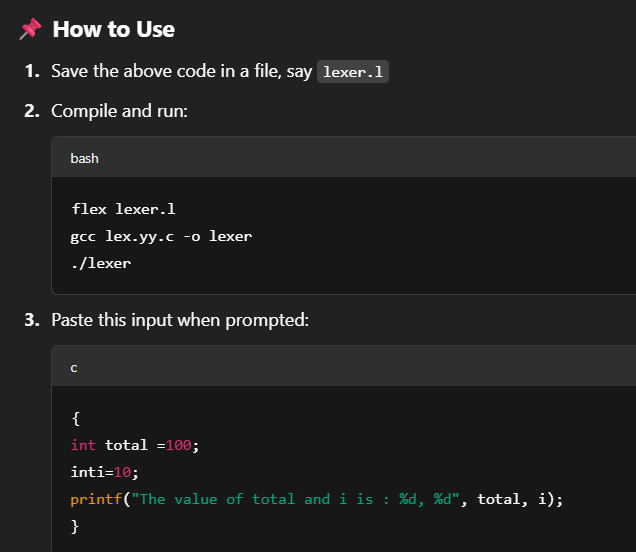
inti=10;

printf("The value of total and i is : %d, %d", total, i);

}







**++++++++++++++++++++++++++++++++++++**

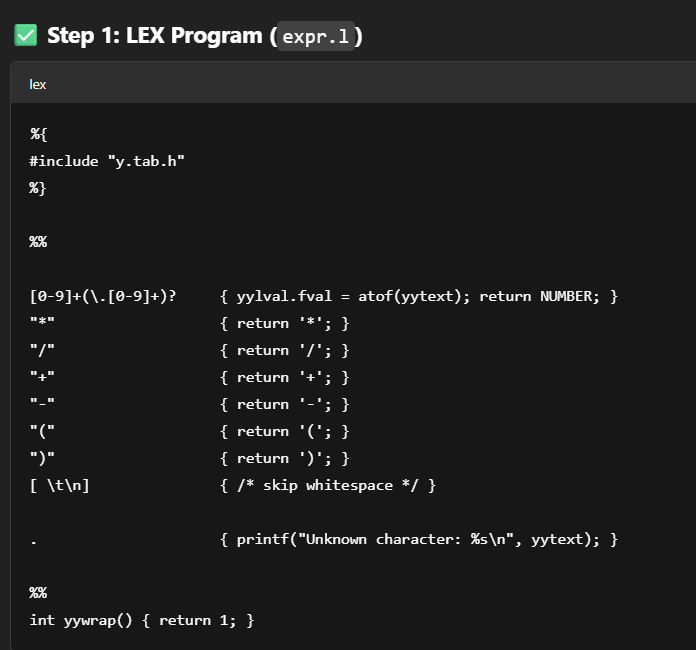
**(32)**

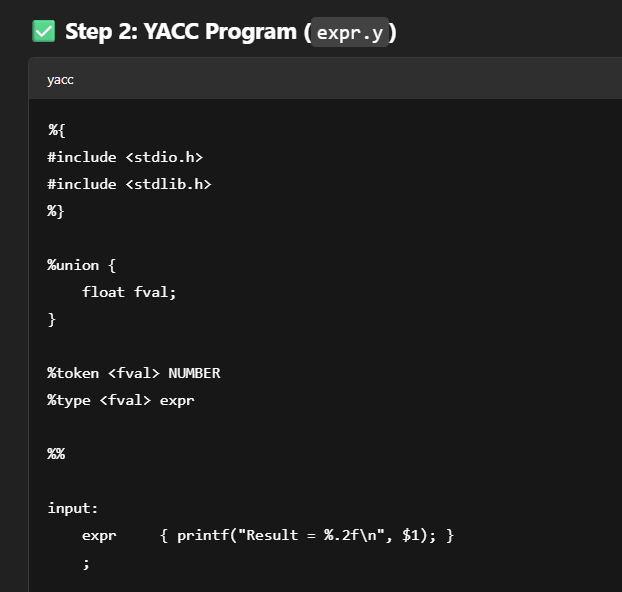
**(A)10.**Write a program to evaluate a given arithmetic expression using YACC specification.

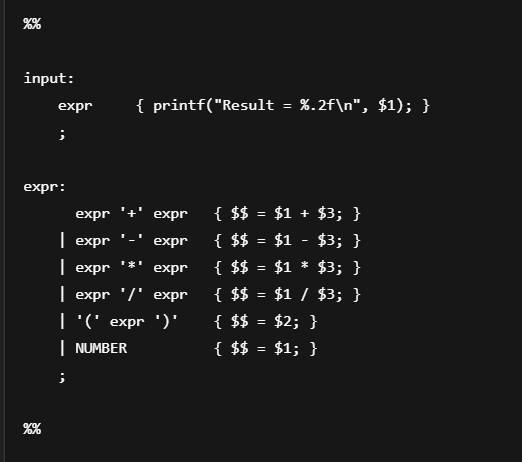
**INPUT**

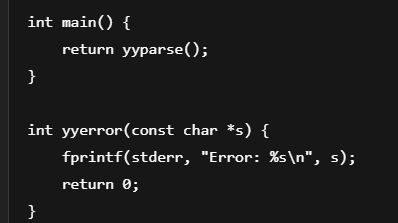
0.33\*12-4-4+(3\*2)

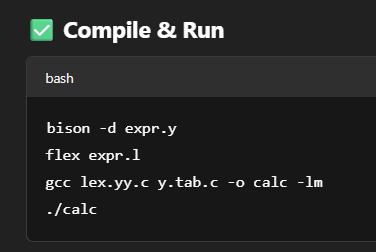
🡪 Here's a **simple YACC (Bison)** and **LEX (Flex)** program that evaluates arithmetic expressions













**++++++++++++++++++++++++++++++++++++**

**(33)**

**(B)10.**Write a program to evaluate a given arithmetic expression using YACC specification.

**INPUT**

(78-44\*4)/3\*2

**++++++++++++++++++++++++++++++++++++**

**(34)**

**(C)10.**Write a program to evaluate a given arithmetic expression using YACC specification.

**INPUT**

98/44-(3\*8)\*4\*2

**++++++++++++++++++++++++++++++++++++**

**(35)**

**(D)10.**Write a program to evaluate a given arithmetic expression using YACC specification.

**INPUT**

1.44+22.4-12.8+11\*3

**++++++++++++++++++++++++++++++++++++**

**(36)**

**(A)11.**Write a program to evaluate a given variable name using YACC specification.

**SAMPLE INPUT**

1. pune
2. PUNE
3. Pune1
4. pUNE\_2

**++++++++++++++++++++++++++++++++++++**

**(37)**

**(B)11**.Write a program to convert small case letters to UPPER case or vise versa using YACC specification.

**SAMPLE INPUT**

1. **Pune – pUNE**
2. **PUNE –pune**

**++++++++++++++++++++++++++++++++++++**

**(38)**

**(C)11.**Write a program to evaluate a given built-in functions using YACC specification.

**INPUT**

1.u= sqrt(36)

2. v = strlen(“pune”)

**++++++++++++++++++++++++++++++++++++**

**(39)**

**(D)11.**Write a program to evaluate a given built-in functions using YACC specification.

**INPUT**

u= sin(12)+cos(12)

**++++++++++++++++++++++++++++++++++++**

**(40)**

**(E)11.**Write a program to evaluate a given built-in functions using YACC specification.

**INPUT**

p= pow(3,2) / log (24)

**++++++++++++++++++++++++++++++++++++**

**(41)**

**(A)12.**Write a program to generate three address code for the given simple expression.

**INPUT**

w = u\*u - u\*v+ v\*v

**++++++++++++++++++++++++++++++++++++**

**(42)**

**(B)12.**Write a program to generate three address code for the given simple expression.

**INPUT**

y=x\*x + w-v / r+r

**++++++++++++++++++++++++++++++++++++**

**(43)**

**(C)12**.Write a program to generate three address code for the given simple expression.

**INPUT**

w = u\*u - u\*v+ v\*v

**++++++++++++++++++++++++++++++++++++**

**(44)**

**(D)12**.Write a program to generate three address code for the given simple expression.

**INPUT**

t = o\*a - o\*b+ o\*c

**++++++++++++++++++++++++++++++++++++**

**(45)**

**(E)12.** Write a program to generate three address code for the given simple expression.

**INPUT**

t = j / k – y / u - i

**++++++++++++++++++++++++++++++++++++**

**(46)**

**(F)12.**Write a program to generate three address code for the given simple expression.

**INPUT**

a = m \* n - o – p / q

**++++++++++++++++++++++++++++++++++++**

**(47)**

**(G)12.**Write a program to generate three address code for the given simple expression.

**INPUT**

a = f ^ r – u \* f \* t – p

**++++++++++++++++++++++++++++++++++++**

**(48)**

**(H)12.**Write a program to generate three address code for the given simple expression.

**INPUT**

a = ( b\*b + c\*c ) \* (p – q – r)

**++++++++++++++++++++++++++++++++++++**