**RAMANUJAN COLLEGE DELHI UNIVERSITY**



**SYSTEM PROGRAMMING PRACTICALS**

**NAME: ADARSH KUMAR**

**COURSE: B.Sc. (H) computer science**

**SEMESTER: 5th**

**UNIVERSITY ROLL NO.: 20020570002**

**SUBMITTED TO: MR. SAHIL PATHAK**

**SIGNATURE**

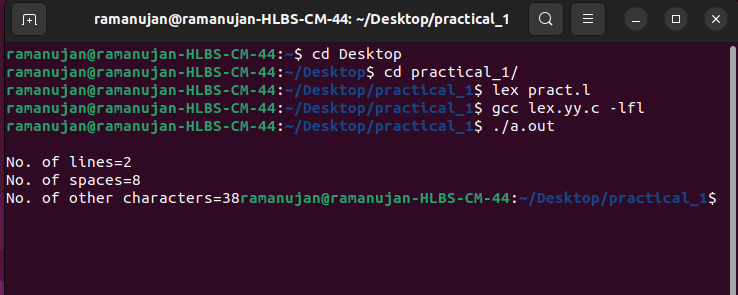
**INDEX**

|  |  |  |
| --- | --- | --- |
| **SNO.** | **PRACTICALS** | |
| **1.** | **1. Write a Lex program to count the number of lines and characters in the input file** | |
| **2.** | **2. Write a Lex program that implements the Caesar cipher: it replaces every letter with the one three letters after in in alphabetical order, wrapping around at Z. e.g., a is replaced by d, b bye, and so on z by c.** | |
| **3.** | **3. Write a Lex program that finds the longest word (defined as a contiguous string of upper- and lower-case letters) in the input.** | |
| **4.** | **4. Write a Lex program that distinguishes keywords, integers, floats, identifiers, operators, and comments in any simple programming language.** | |
| **5.** | **5. Write a Lex program to count the number of identifiers in a C file.** | |
| **6.** | **6. Write a Lex program to count the number of words, characters, blank spaces and lines in a C file.** | |
| **7.** | **7. Write a Lex specification program that generates a C program which takes a string “abcd” and prints the following output. abcd abc ab a** | |
| **8.** | **8. A program in Lex to recognize a valid arithmetic expression.** | |
| **9.** | **9.Write a YACC program to find the validity of a given expression (for operators + - \* and /)** | |
| **10.** | **10. A Program in YACC which recognizes a valid variable which starts with letter followed by a digit. The letter should be in lowercase only.** | |
| **11.** | **11. A Program in YACC to evaluate an expression (simple calculator program for addition and subtraction, multiplication, division).** | |
| **12.** | **12. Program in YACC to recognize the strings “ab”, “aabb”,” aaabbb” … of the language (𝑎 𝑛 𝑏 𝑛, n>=1).** | |
| **13.** | | **13. Program in YACC to recognize the language (𝑎 𝑛𝑏, n>=10). (Output to say input is valid or not)** |

**ANS 1.**

|  |
| --- |
| %{ |
|  | #include<stdio.h> |
|  | int lc=0, sc=0, tc=0, ch=0; /\*Global variables\*/ |
|  | %} |
|  |  |
|  | /\*Rule Section\*/ |
|  | %% |
|  | \n lc++; //line counter |
|  | ([ ])+ sc++; //space counter |
|  | \t tc++; //tab counter |
|  | . ch++; //characters counter |
|  | %% |
|  |  |
|  | int main() |
|  | { |
|  | // The function that starts the analysis |
|  | yyin=fopen("abc.txt","r"); |
|  | yylex(); |
|  |  |
|  | printf("\nNo. of lines=%d", lc); |
|  | printf("\nNo. of spaces=%d", sc); |
|  |  |
|  | printf("\nNo. of other characters=%d", ch); |
|  |  |
|  | } |

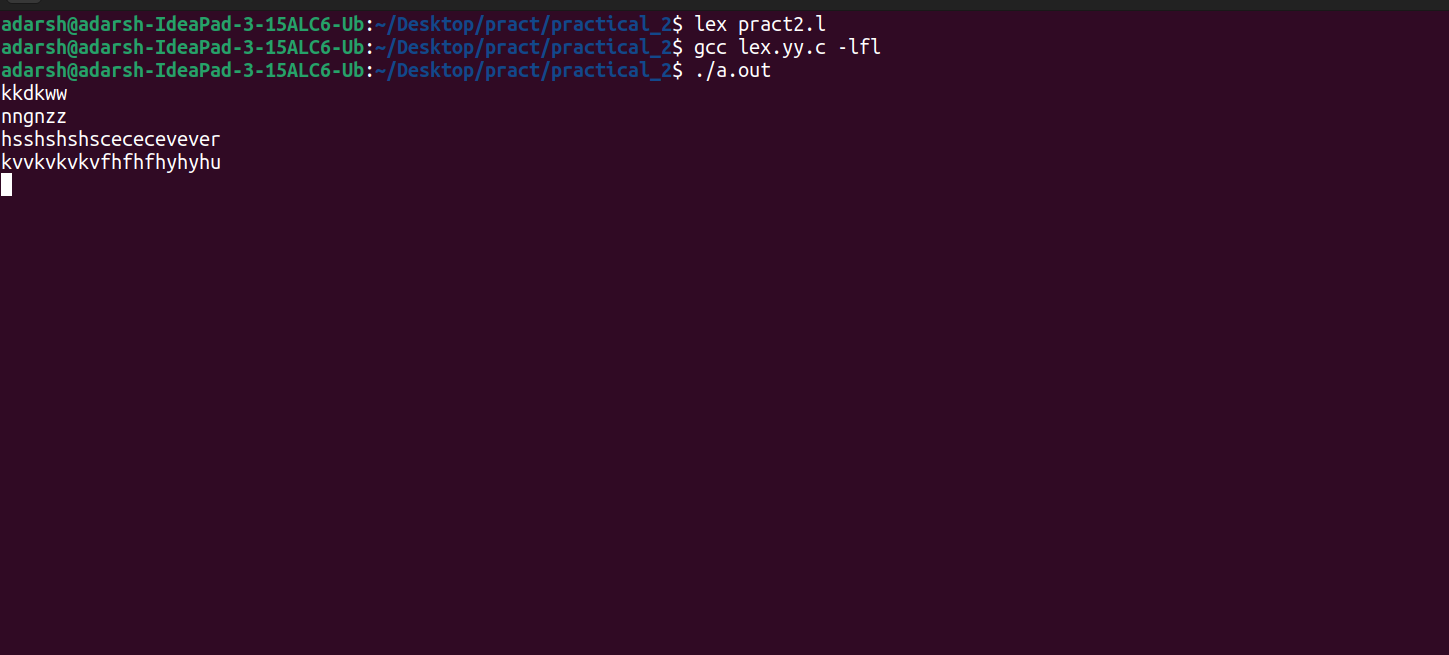
**OUTPUT:**



**ANS 2.**

|  |
| --- |
| %{ |
|  |  |
|  | %} |
|  | %% |
|  | [A-Wa-w] {printf("%c",yytext[0]+3);} |
|  | [X-Zx-z] {printf("%c",yytext[0]-23);} |
|  | %% |
|  | int main() |
|  | { |
|  | //yyin=fopen("bbc.txt","r"); |
|  | //yyout=fopen("kbc.txt","w"); |
|  | yylex(); |
|  |  |
|  | } |

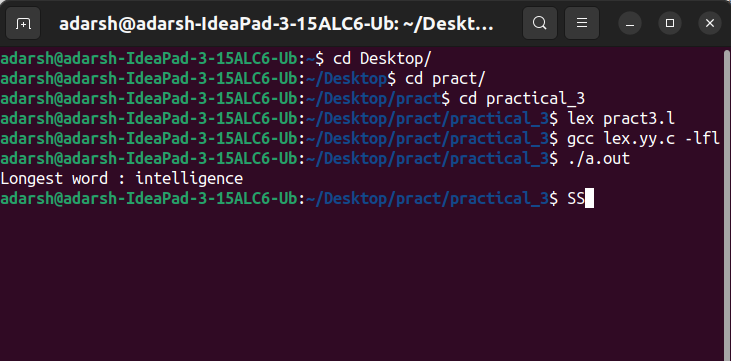
**OUTPUT:**



**ANS 3.**

|  |
| --- |
| %{ |
|  | #include<stdio.h> |
|  | #include<strings.h> |
|  | // initialising length |
|  | int length=0; |
|  | // char array for storing longest word |
|  | char longestword[50]; |
|  | %} |
|  |  |
|  | %% |
|  | [A-Za-z0-9]+ { if (yyleng > length) { |
|  |  |
|  | length=yyleng; |
|  | // strcpy function to copy current word in yytxt in longest |
|  | strcpy(longestword,yytext); |
|  | } |
|  | } |
|  | "." return 1; |
|  | %% |
|  | int main() |
|  | { |
|  | yyin=fopen("tbc.txt","r"); |
|  |  |
|  | yylex(); |
|  | printf("Longest word : %s\n",longestword); |
|  | //printf("Length of Longest word : %s\n",length); |
|  |  |
|  | return 0; |
|  | } |
|  | int yywrap(){ |
|  | return 1; |
|  | } |

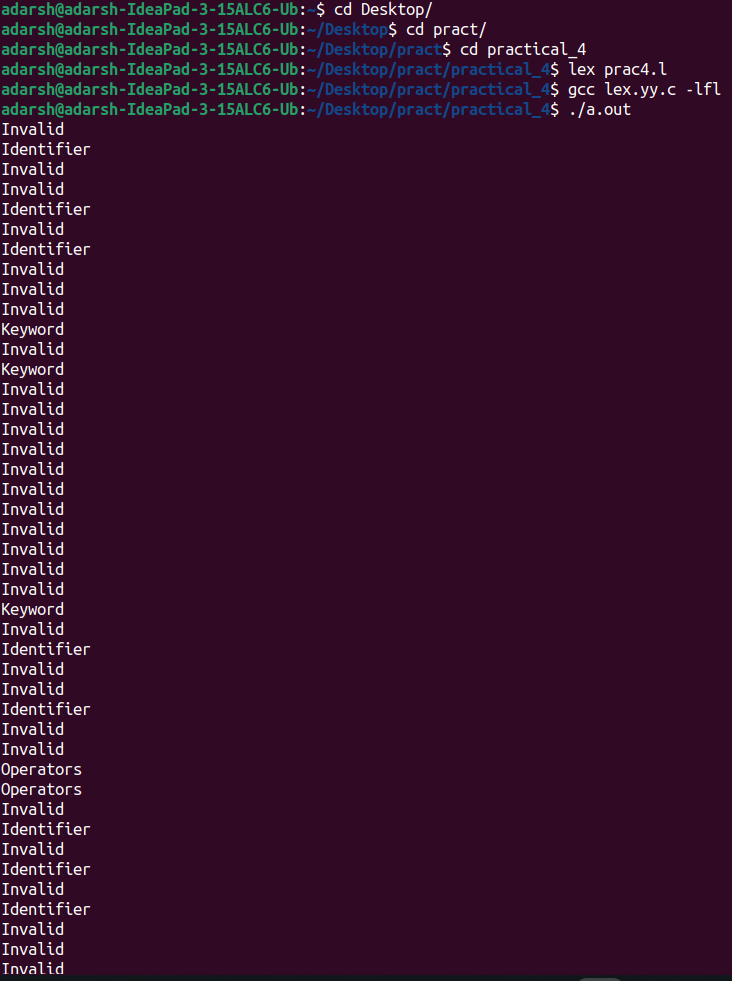
**OUTPUT:**



**ANS 4.**

|  |
| --- |
|  |
| %{ |
|  |  |
|  | %} |
|  | %% |
|  | [0-9]\* {printf("Integer\n");} |
|  | [0-9]+\.[0-9]+ {printf("Float\n"); } |
|  | int|float|if|else|printf|main|exit|switch {printf("Keyword\n");} |
|  | [+|\*|/|%|&] {printf("Operators\n");} |
|  | "-" {printf("Operators\n");} |
|  | "/\*".\*"\*/" {printf("comment\n");} |
|  | [\_a-zA-Z][\_a-zA-Z0-9]{0,30} {printf("Identifier\n");} |
|  | . {printf("Invalid\n");} |
|  | %% |
|  | int main() |
|  | { |
|  | yyin=fopen("code.c","r"); |
|  | yyout=fopen("kmd.txt","w"); |
|  | yylex(); |
|  |  |
|  | } |
|  |  |

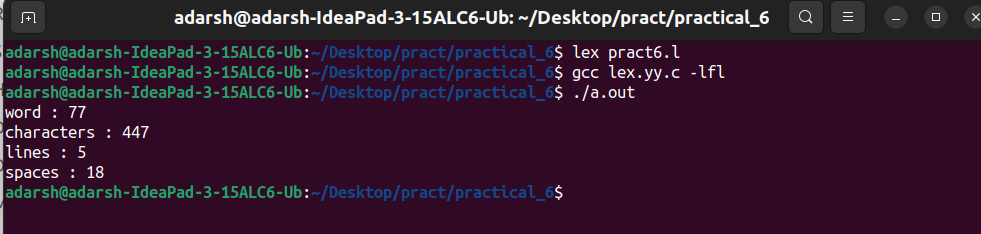
**OUTPUT:**



**ANS 5.**

|  |
| --- |
|  |
| %{ |
|  | #include<stdio.h> |
|  | int word=0,character=0,space=0,lines=0; |
|  | %} |
|  | %% |
|  | [A-Za-z|0-9]+ {word++;character=character+strlen(yytext);} |
|  | . {character++;} |
|  | \n {lines++;character++;} |
|  | [ \n\t\r]+ {space++;} |
|  | %% |
|  | int main(int agrc,char \*\*argv) |
|  | { |
|  | yyin=fopen("pla.txt","r"); |
|  | yylex(); |
|  | printf("word : %d\n",word); |
|  | printf("characters : %d\n",character); |
|  | printf("lines : %d\n",lines); |
|  | printf("spaces : %d\n",space); |
|  | } |
|  |  |

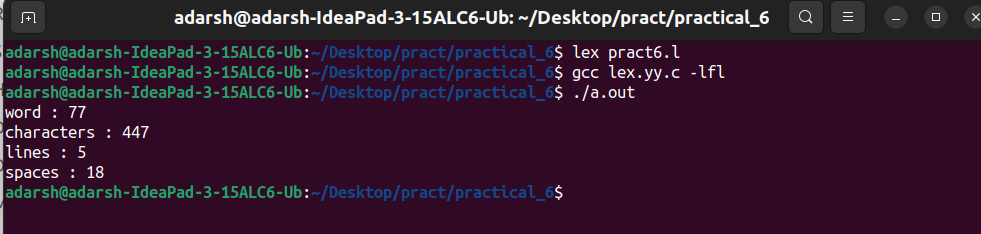
**OUTPUT:**



**ANS 6.**

|  |
| --- |
|  |
| %{ |
|  | #include<stdio.h> |
|  | int word=0,character=0,space=0,lines=0; |
|  | %} |
|  | %% |
|  | [A-Za-z|0-9]+ {word++;character=character+strlen(yytext);} |
|  | . {character++;} |
|  | \n {lines++;character++;} |
|  | [ \n\t\r]+ {space++;} |
|  | %% |
|  | int main(int agrc,char \*\*argv) |
|  | { |
|  | yyin=fopen("pla.txt","r"); |
|  | yylex(); |
|  | printf("word : %d\n",word); |
|  | printf("characters : %d\n",character); |
|  | printf("lines : %d\n",lines); |
|  | printf("spaces : %d\n",space); |
|  | } |
|  |  |

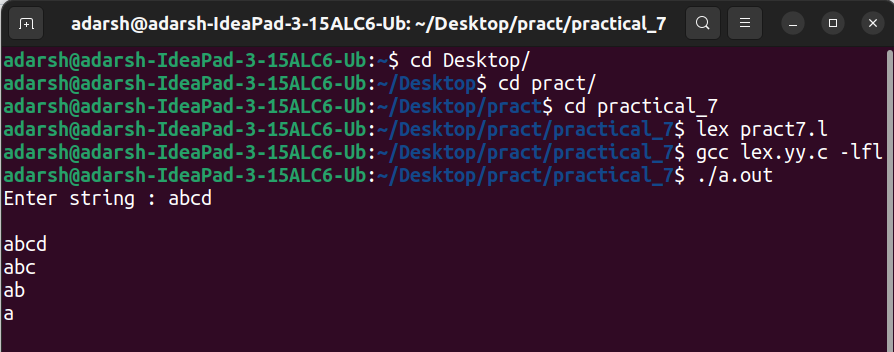
**OUTPUT:**



**ANS 7.**

|  |
| --- |
|  |
| %{ |
|  | %} |
|  | %% |
|  | [A-Za-z]+ {int len=yyleng; |
|  | int i=len; |
|  | printf("\n"); |
|  | while(i>=0) |
|  | { |
|  | int j=0; |
|  | while(j<i) |
|  | { |
|  | printf("%c",yytext[j]); |
|  | j++; |
|  | } |
|  | printf("\n"); |
|  | i--; |
|  | } |
|  | } |
|  | %% |
|  | int main() |
|  | { |
|  | printf("Enter string : "); |
|  | yylex(); |
|  | } |
|  |  |

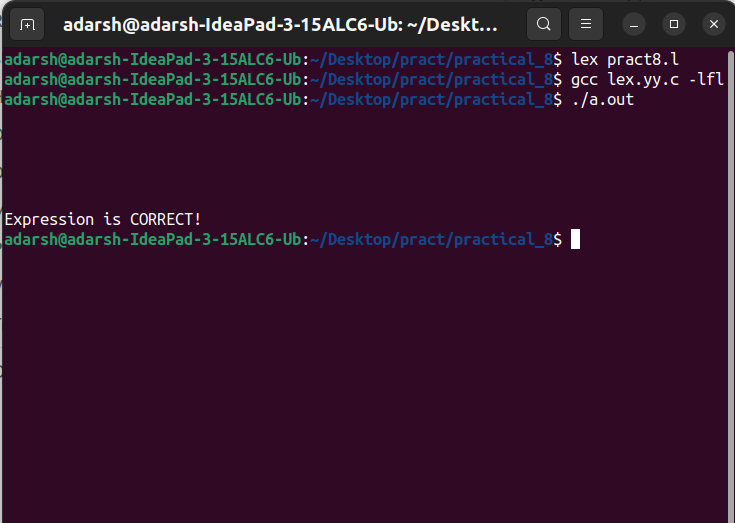
**OUTPUT:**



**ANS 8.**

|  |
| --- |
| %{ |
|  | #include<strings.h> |
|  | int opcount=0,intcount=0,check=1,top=0; |
|  | %} |
|  | %% |
|  | ['('] {check=0;} |
|  | [')'] {check=1;} |
|  | [+|\*|/|-] {opcount++;} |
|  | [0-9]+ {intcount++;} |
|  | . {printf("Invalid Input only digits and +|-|\*|/ is valid\n");} |
|  | %% |
|  | int main() |
|  | { |
|  |  |
|  | yyin=fopen("abd.txt","r"); |
|  | yylex(); |
|  | if(intcount=opcount+1) |
|  | { |
|  | if(check==1) |
|  | { |
|  | printf("Expression is CORRECT!\n"); |
|  | } |
|  | else{ |
|  | printf("')' bracket missing from expression\n"); |
|  | } |
|  | } |
|  | else{ |
|  | printf("Expression is INCORRECT!\n"); |
|  | } |
|  | } |

**OUTPUT:**



**ANS 9.**

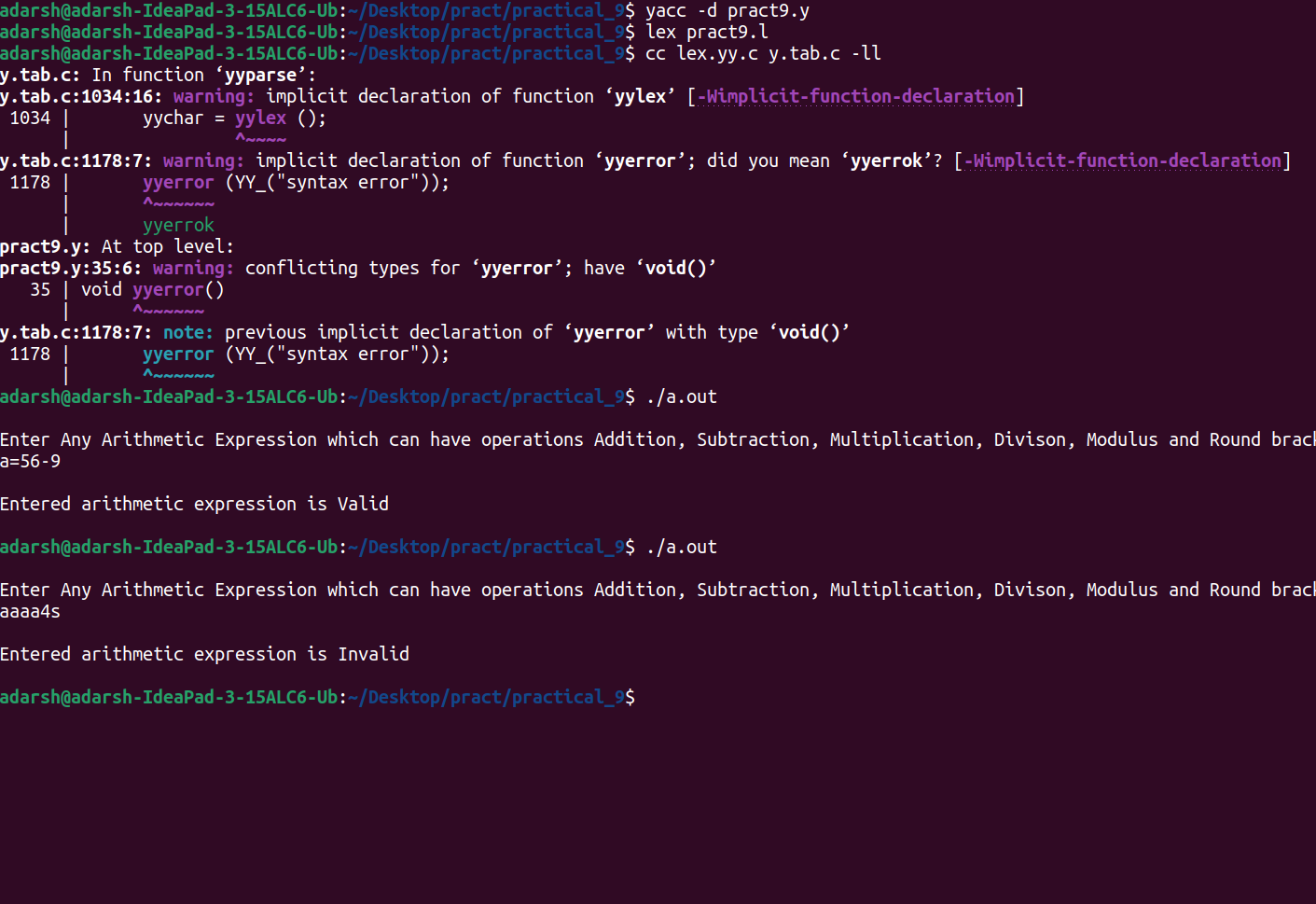
**Lex program:**

|  |
| --- |
| %{ |
|  | #include<stdio.h> |
|  | #include "y.tab.h" |
|  | %} |
|  |  |
|  | %% |
|  | [a-zA-Z]+ return VARIABLE; |
|  | [0-9]+ return NUMBER; |
|  | [\t] ; |
|  | [\n] return 0; |
|  | . return yytext[0]; |
|  | %% |
|  | int yywrap() |
|  | { |
|  | return 1; |
|  | } |

Yacc program:

|  |
| --- |
| %{ |
|  | #include<stdio.h> |
|  | %} |
|  | %token NUMBER |
|  | %token VARIABLE |
|  |  |
|  | %left '+' '-' |
|  | %left '\*' '/' '%' |
|  | %left '(' ')' |
|  |  |
|  | %% |
|  |  |
|  | S: VARIABLE'='E { |
|  | printf("\nEntered arithmetic expression is Valid\n\n"); |
|  | return 0; |
|  | } |
|  | E:E'+'E |
|  | |E'-'E |
|  | |E'\*'E |
|  | |E'/'E |
|  | |E'%'E |
|  | |'('E')' |
|  | | NUMBER |
|  | | VARIABLE |
|  | ; |
|  |  |
|  | %% |
|  |  |
|  | void main() |
|  | { |
|  | printf("\nEnter Any Arithmetic Expression which can have operations Addition, Subtraction, Multiplication, Divison, Modulus and Round brackets:\n"); |
|  | yyparse(); |
|  | } |
|  |  |
|  | void yyerror() |
|  | { |
|  | printf("\nEntered arithmetic expression is Invalid\n\n"); |
|  |  |
|  | } |

**OUTPUT:**



**ANS 10.**

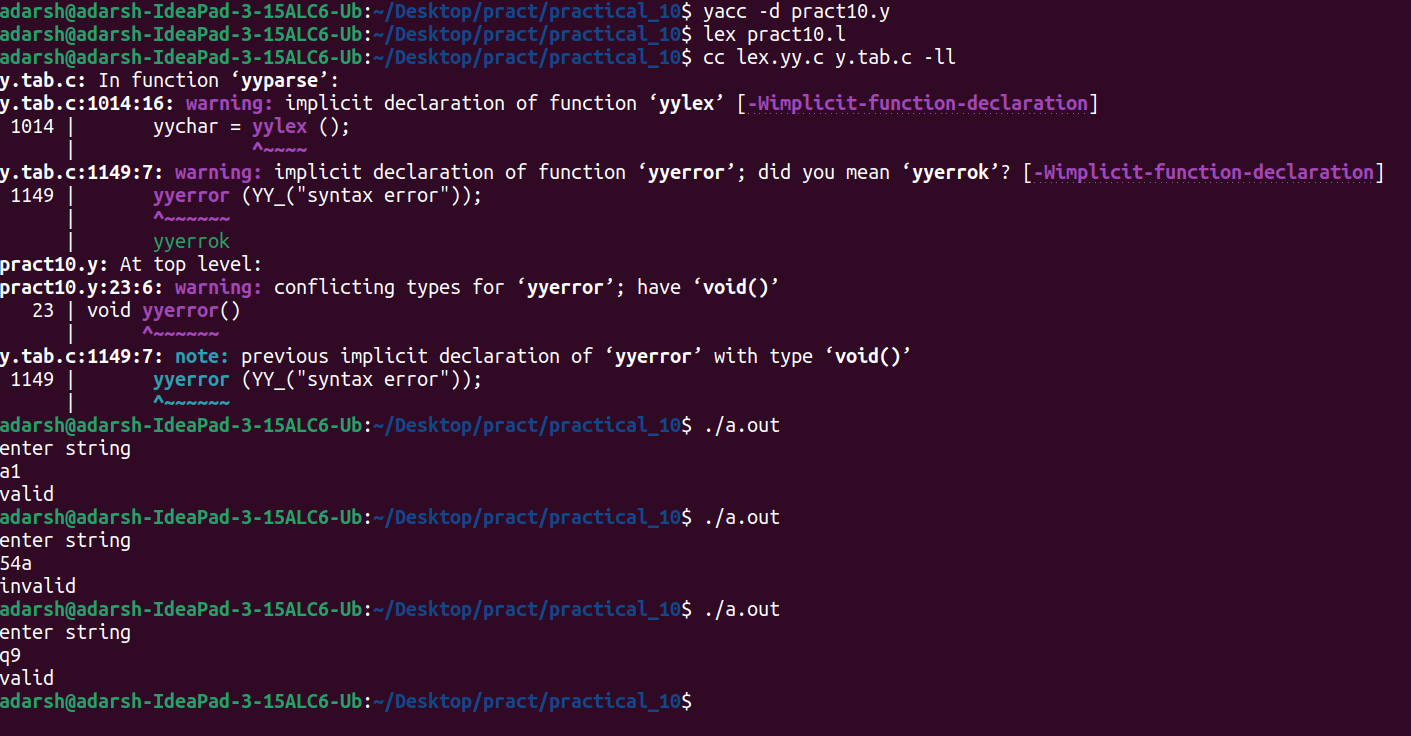
**Lex program:**

|  |
| --- |
| %{ |
|  | #include "y.tab.h" |
|  | %} |
|  | %% |
|  | [0-9]+ {return DIGIT;} |
|  | [a-z]+ {return LETTER;} |
|  | [ \t] {;} |
|  | \n { return 0;} |
|  | . {return yytext[0];} |
|  | %% |

**Yacc program:**

|  |
| --- |
| %{ |
|  | #include<stdio.h> |
|  | #include<stdlib.h> |
|  | %} |
|  | %token DIGIT LETTER |
|  | %% |
|  | stmt:A |
|  | ; |
|  | A: LETTER B |
|  | ; |
|  | B: LETTER B |
|  | | DIGIT B |
|  | | LETTER |
|  | | DIGIT |
|  | ; |
|  | %% |
|  | void main(){ |
|  | printf("enter string \n"); |
|  | yyparse(); |
|  | printf("valid \n"); |
|  | exit(0); |
|  | } |
|  | void yyerror() |
|  | { |
|  | printf("invalid \n"); |
|  | exit(0); |
|  | } |

**OUTPUT:**



**ANS 11.**

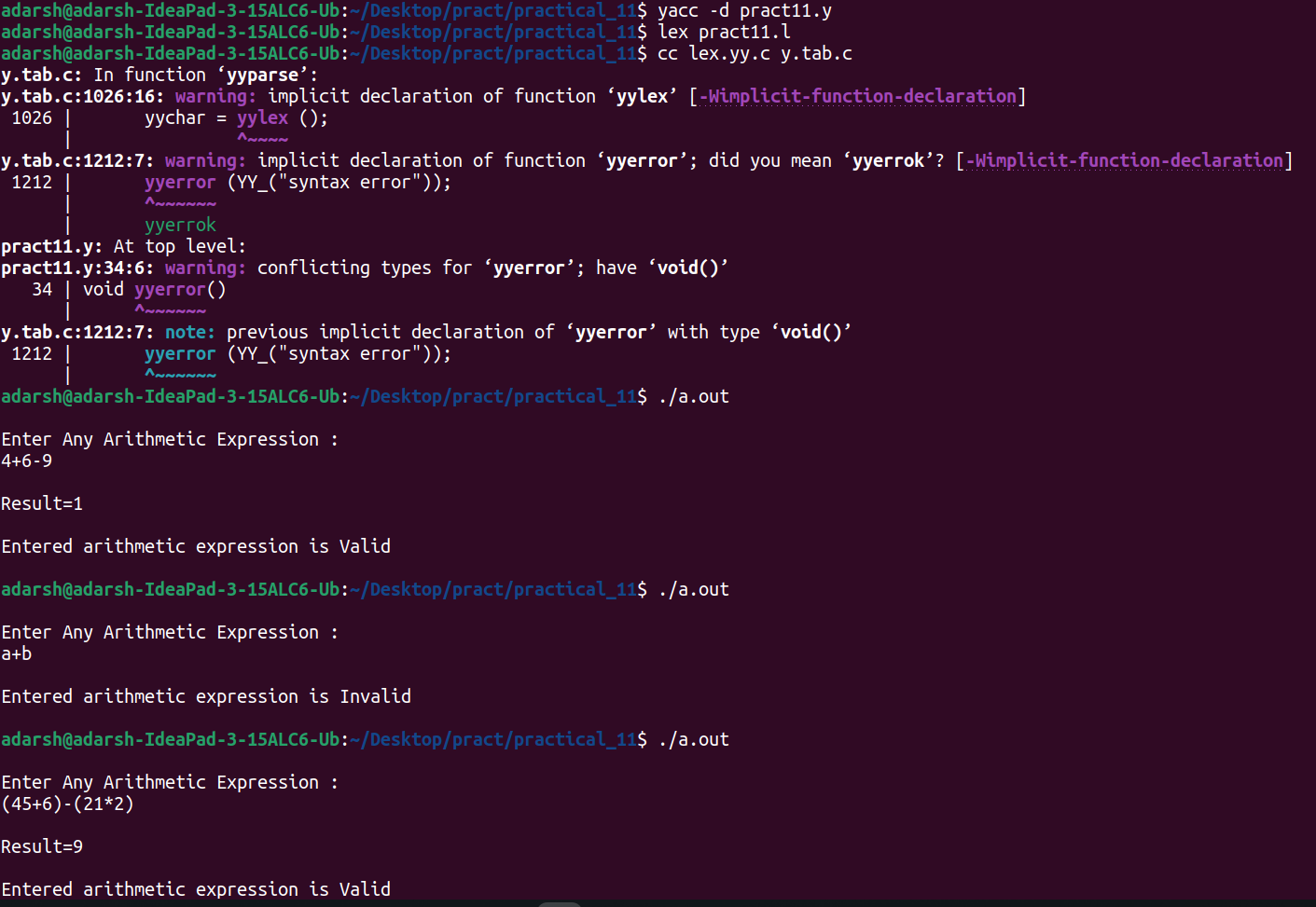
**Lex program:**

|  |
| --- |
| %{ |
|  | #include<stdio.h> |
|  | #include "y.tab.h" |
|  | extern int yylval; |
|  | %} |
|  |  |
|  | %% |
|  | [0-9]+ { |
|  | yylval=atoi(yytext); |
|  | return NUMBER; |
|  | } |
|  | [\t] ; |
|  | [\n] return 0; |
|  | . return yytext[0]; |
|  | %% |
|  | int yywrap() |
|  | { |
|  | return 1; |
|  | } |

**Yaac program:**

|  |
| --- |
| %{ |
|  | #include<stdio.h> |
|  | int flag=0; |
|  |  |
|  | %} |
|  | %token NUMBER |
|  |  |
|  | %left '+' '-' |
|  | %left '\*' '/' '%' |
|  | %left '(' ')' |
|  | %% |
|  | ArithmeticExpression: E{ |
|  | printf("\nResult=%d\n",$$); |
|  | return 0; |
|  | } |
|  | E:E'+'E {$$=$1+$3;} |
|  | |E'-'E {$$=$1-$3;} |
|  | |E'\*'E {$$=$1\*$3;} |
|  | |E'/'E {$$=$1/$3;} |
|  | |E'%'E {$$=$1%$3;} |
|  | |'('E')' {$$=$2;} |
|  | | NUMBER {$$=$1;} |
|  | ; |
|  | %% |
|  |  |
|  | void main() |
|  | { |
|  | printf("\nEnter Any Arithmetic Expression :\n"); |
|  | yyparse(); |
|  | if(flag==0) |
|  | printf("\nEntered arithmetic expression is Valid\n\n"); |
|  |  |
|  | } |
|  | void yyerror() |
|  | { |
|  | printf("\nEntered arithmetic expression is Invalid\n\n"); |
|  | flag=1; |
|  | } |

**OUTPUT:**



**ANS 12.**

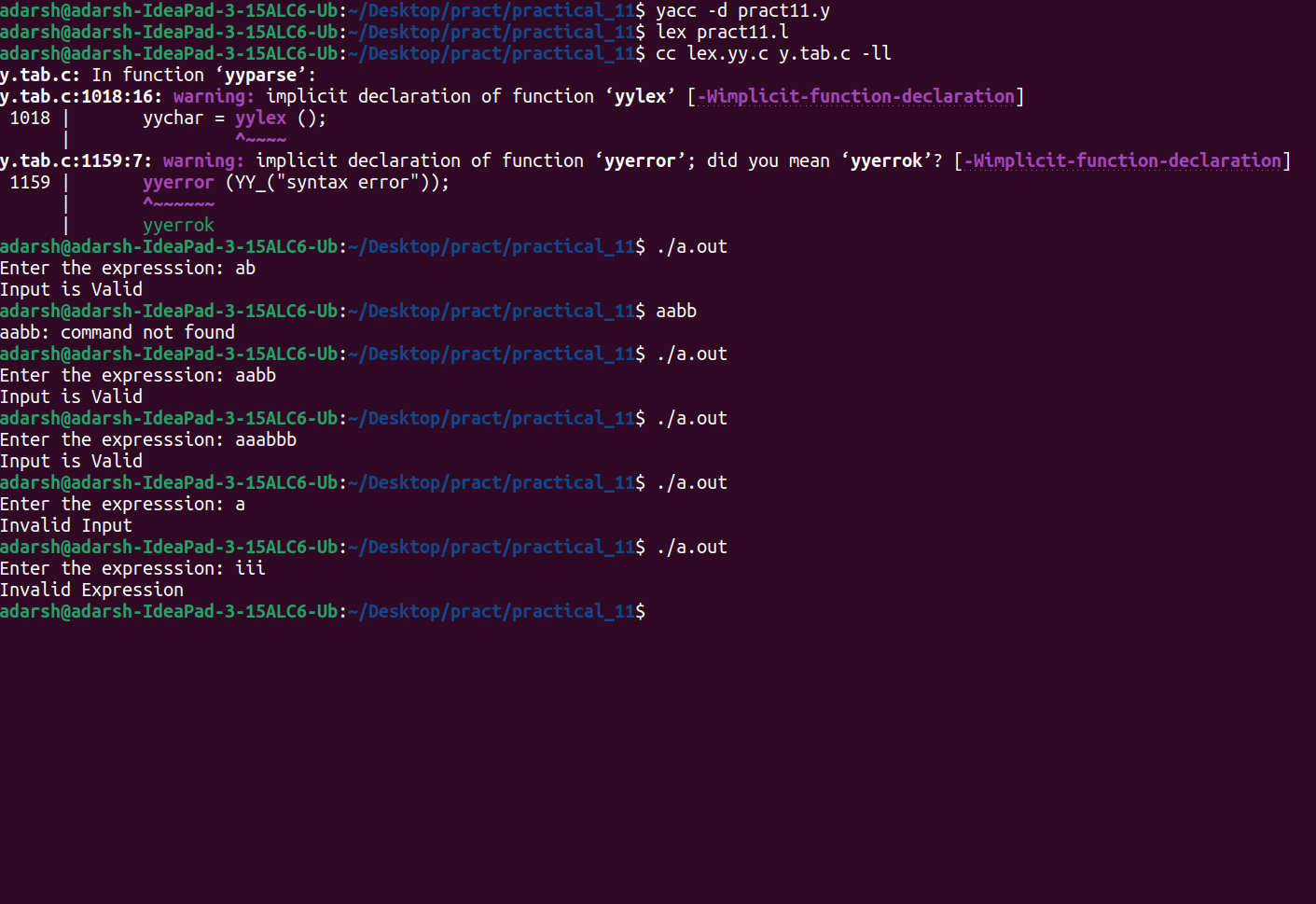
**Lex program:**

|  |
| --- |
|  |
| %{ |
|  | #include "y.tab.h" |
|  | %} |
|  | alpha [Aa] |
|  | beta [Bb] |
|  | newline [\n] |
|  | %% |
|  | {alpha} { return alpha ;} |
|  | {beta} {return beta;} |
|  | {newline} { return newline ;} |
|  | . { printf("Invalid Expression\n");exit(0); } |
|  | %% |
|  |  |

**Yaac program:**

|  |
| --- |
| %{ |
|  | #include<stdio.h> |
|  | #include<stdlib.h> |
|  | #include<strings.h> |
|  | %} |
|  | %token alpha beta newline |
|  | %% |
|  | line : term newline {printf("Input is Valid\n"); exit(0);}; |
|  | term: alpha term beta | ; |
|  | %% |
|  |  |
|  | int yyerror(char \*msg) |
|  | { |
|  | printf("Invalid Input\n"); |
|  | exit(0); |
|  | } |
|  |  |
|  | int main () |
|  | { |
|  | printf("Enter the expresssion: "); |
|  | yyparse(); |
|  | } |

**OUTPUT:**



**ANS 13.**

**Lex program:**

|  |
| --- |
| %{ |
|  | #include "y.tab.h" |
|  | %} |
|  | alpha [a]{10,} |
|  | beta [b] |
|  | newline [\n] |
|  | %% |
|  | {alpha} { return alpha ;} |
|  | {beta} {return beta;} |
|  | {newline} { return newline ;} |
|  | . { printf("Invalid Expression\n");exit(0); } |
|  | %% |

**Yaac program:**

|  |
| --- |
| %{ |
|  | #include<stdio.h> |
|  | #include<stdlib.h> |
|  | #include<strings.h> |
|  | %} |
|  | %token alpha beta newline |
|  | %% |
|  | line : term beta newline {printf("Input is Valid\n"); exit(0);}; |
|  | term: alpha term |; |
|  | %% |
|  |  |
|  | int yyerror(char \*msg) |
|  | { |
|  | printf("Invalid Input\n"); |
|  | exit(0); |
|  | } |
|  |  |
|  | int main () |
|  | { |
|  | printf("Enter the expresssion: "); |
|  | yyparse(); |
|  | } |

**OUTPUT:**

