Ram Lal Anand College

(University of Delhi)

Department of Computer Science

Practical File

Session- JULY-DECEMBER 2022

Name of Program/Course: - B.Sc. (H) Computer Science

Semester: - V

Title of practical /Name of the Paper:- SYSTEM PROGRAMMING

Paper Code: - 32347501

Name of the Student: HARSH JAISWAL

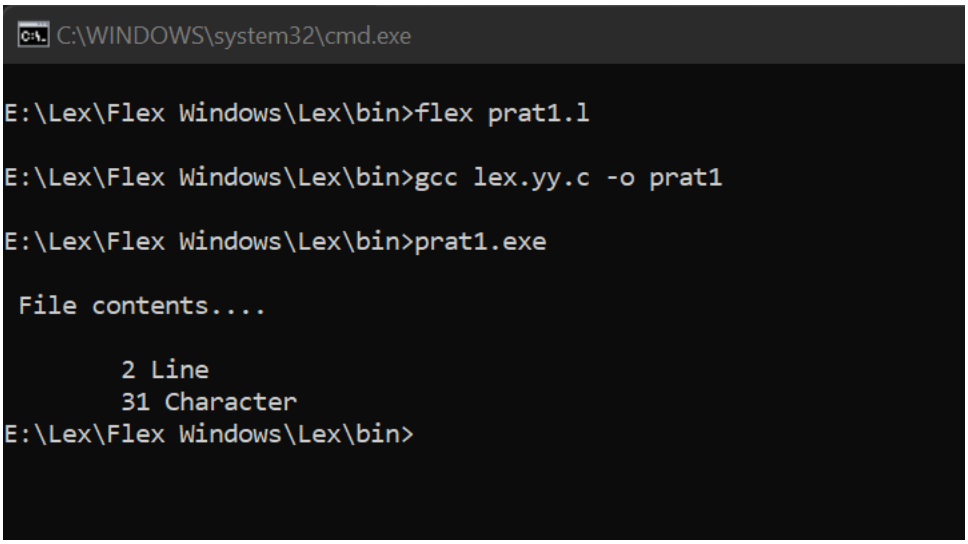
Examination Roll No: - 20058570011

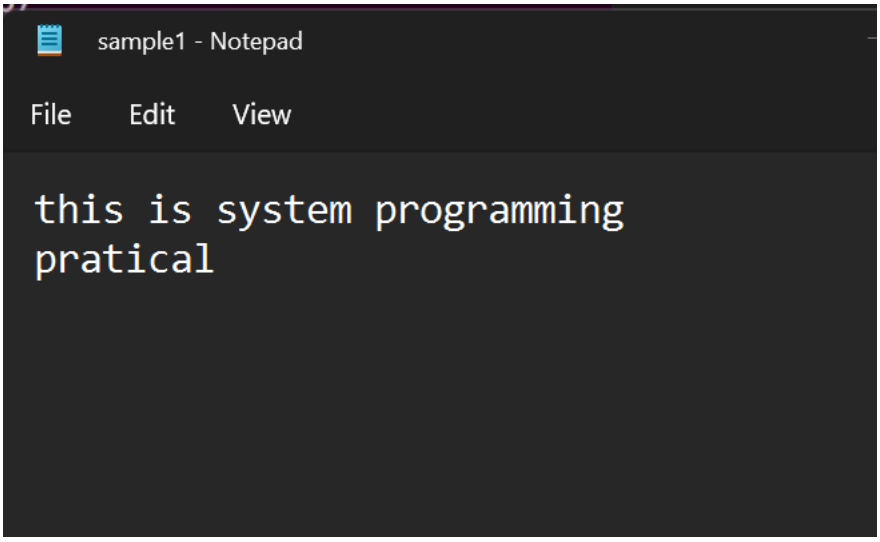
Harsh (4016)

Q1 Write a Lex program to count the number of lines and characters in the input file.

```
%{
        #include<stdio.h>
        int lines =0,lcharacter=0;
%}

%%
\n {lines++;}
[A-Za-z] lcharacter++;

%%
int main()
{

       yyin = fopen("sample1.txt","r");
          yylex();
          printf("\n File contents....\n");
          printf("\n\t%d Line ",lines);
          printf("\n\t%d Character ",lcharacter);

        return 0;
}
int yywrap()
{
```

```
return 1;
}
```





Ques 2 Write a Lex program that implements the Caesar cipher: it replaces every letter with the one three letters after in alphabetical order, wrapping around at Z. e.g. a is replaced by d, b by e, and so on z by c.

```
% {
        #include<stdio.h>
%}
```

```
%%
[A-Wa-w] {printf("%c",yytext[0]+3);}
[X-Zx-z] {printf("%c",yytext[0]-23);}
%%
int main()
{

      printf("Enter your text \n");
       yylex();
        return 0;
}
```

```
E:\Lex\Flex Windows\Lex\bin>flex prt2.l

E:\Lex\Flex Windows\Lex\bin>gcc lex.yy.c -o prt2

E:\Lex\Flex Windows\Lex\bin>prt2.exe
Enter your text
XYZ
ABC
Hello Wolrd
Khoor Zroug
system programming
vbvwhp surjudpplqj
```

Ques 3 Write a Lex program that finds the longest word (defined as a contiguous string of upper- and lower-case letters) in the input.

```
%{
```

```
        #include<stdio.h>
        #include<string.h>
        int count =0;
        char longest[68];
%}

%%
[A-Za-z0-9]+ { if (yyleng >count) {
                count = yyleng;
                    strcpy(longest,yytext);
                    }
          }
%%
int main()
{
   yyin = fopen("sample1.txt","r");
   yylex();
   printf("longest word is : %s\n",longest);

        return 0;
}
int yywrap()
{
  return 1;
}
```

Harsh (4016)

```
E:\Lex\Flex Windows\Lex\bin>flex prat3.l

E:\Lex\Flex Windows\Lex\bin>gcc lex.yy.c -o prat3

E:\Lex\Flex Windows\Lex\bin>prat3.exe


longest word is : programming

E:\Lex\Flex Windows\Lex\bin>
```

sample1 - Notepad

File Edit View

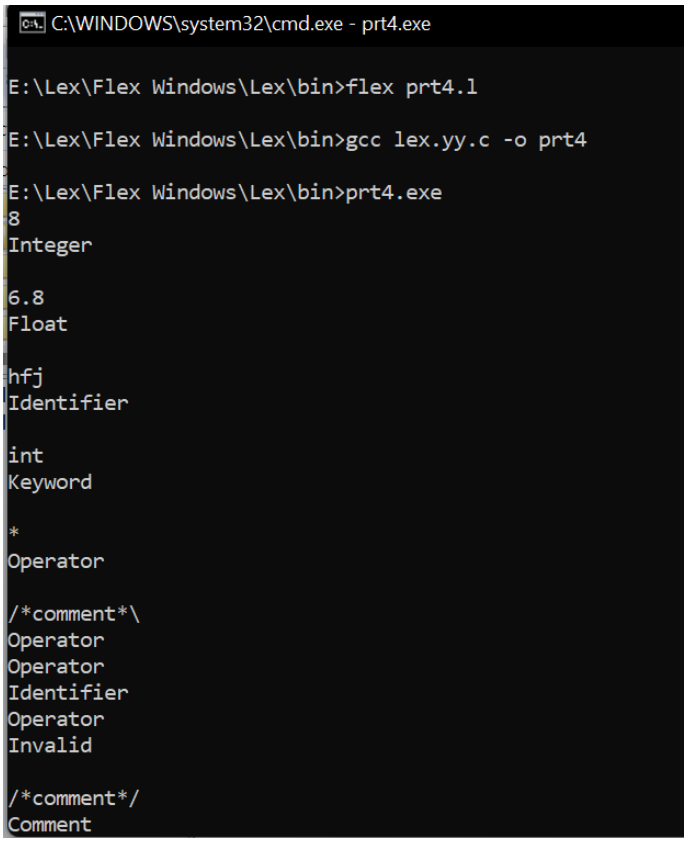
```
this is system programming
pratical
```

Q4 Write a Lex program that distinguish keywords ,integers ,floats ,identifiers ,operators and comments in any program

```
%{
        #include<stdio.h>
%}


%%
[0-9]* {printf("Integer\n");}
[0-9]+\.[0-9]+ {printf("Float\n");}
```

Harsh (4016)

```
int|float|if|else|char|printf|main|switch {printf("Keyword\n");}
[+|*|/|%|-|&] {printf("Operator\n");}
"/*".*"*/" {printf("Comment\n");}
[_a-zA-Z][_a-zA-Z0-9]{0,30} {printf("Identifier\n");}
. {printf("Invalid\n");}
%%
int main()
{
      yylex();
         return 0;
}
int yywrap()
{
 return 1;
 }
```



Harsh (4016)

Q5 Write a Lex program to count the number of identifiers in a C file.

```
%{
#include <stdio.h>
int count=0;

%}
WS [ \t\n]*
ID [_a-zA-Z][_a-zA-Z0-9]*
DECLN "int"|"float"|"char"|"short"|"long"|"unsigned"
%x DEFN
%%
{DECLN}  {BEGIN DEFN;}
<DEFN>{WS}{ID}{WS}\,    count++;
<DEFN>{WS}{ID}{WS}\;    count++;
<*>\n ;
<*>.    ;                                                    ;
%%
int main()
{
yyin=fopen("q55.c","r");
yylex();
printf("No of identifiers : %d\n",count);
return 0;
}
```

```
int yywrap(){
return 1;
}
```

```
E:\Lex\Flex Windows\Lex\bin>gcc lex.yy.c -o q5

E:\Lex\Flex Windows\Lex\bin>q5.exe
No of identifiers : 5

E:\Lex\Flex Windows\Lex\bin>
```

q55 - Notepad

File Edit View

```
#include <stdio.h>
int main() {

    int number1,xyz;
    int number2;
    double sum;
    char xyz;
    printf("5 variable");

    return 0;
}
```

Q6 Write a Lex program to count the number of words, characters, blank spaces and lines in a C file.

```
%{
```

```
	#include<stdlib.h>
	int lines=0, blanks=0, chars=0, words=0;
%}

%%
\n { lines++; chars++;}
([a-zA-Z0-9])* { words++; chars += yyleng;}
([ ])+ { blanks++; chars++;}
\t { blanks += 4; chars ++;}
. { chars++;}
%%
int main(int argc, char*argv[]){
	yyin = fopen(argv[1],"r");
	yylex();
	printf("\n'%s' has total\n", argv[1]);
	printf("-> %d new lines\n", lines);
	printf("-> %d blanks\n", blanks);
	printf("-> %d characters\n", chars);
	printf("-> %d words.\n\n", words);
	fclose(yyin);

	return 1;
}

int yywrap(){
```

```
        return 1;
}
```

```
E:\Lex\Flex Windows\Lex\bin>flex prat6.l

E:\Lex\Flex Windows\Lex\bin>gcc lex.yy.c -o prat6

E:\Lex\Flex Windows\Lex\bin>prat6.exe sumOfDigits.c

'sumOfDigits.c' has total
-> 13 new lines
-> 34 blanks
-> 225 characters
-> 30 words.
```

sumOfDigits - Notepad

File Edit View

```
#include <stdio.h>
int main() {

    int number1, number2, sum;

    printf("Enter two integers: ");
    scanf("%d %d", &number1, &number2);

    sum = number1 + number2;

    printf("%d + %d = %d", number1, number2, sum);
    return 0;
}
```

Q 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

```
%{
        #include<stdio.h>
%}
%%
[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();
        return 0;
}
```

```
E:\Lex\Flex Windows\Lex\bin>flex prat7.l

E:\Lex\Flex Windows\Lex\bin>gcc lex.yy.c -o prat7

E:\Lex\Flex Windows\Lex\bin>prat7.exe
Enter string : abcd

abcd
abc
ab
a
```

Q8 A program in Lex to recognize a valid arithmetic expression.

```
%{
#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("q8.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");
}
  return 0;
}
int yywrap()
{
  return 1;
 }
```

```
E:\Lex\Flex Windows\Lex\bin>flex prat8.l

E:\Lex\Flex Windows\Lex\bin>gcc lex.yy.c -o prat8

E:\Lex\Flex Windows\Lex\bin>prat8.exe


Expression is CORRECT!

E:\Lex\Flex Windows\Lex\bin>prat8.exe


')' bracket missing from expression

E:\Lex\Flex Windows\Lex\bin>prat8.exe
```

Harsh (4016)

Question -9 Write a YACC program to find the validity of a given expression (for operators + - * and /)

yaac1.l file

```
%{
        #include<stdio.h>
        #include<stdlib.h>
        #include "yaac1.tab.h"
%}

%%

[\t]+ ;

[0-9]+  { printf("\n %s is a valid number\n",yytext);
        return NUM;}

[a-z_]+[a-z_0-9]*  { printf("\n %s is a valid variable\n",yytext);
        return VAR;}

[+] {printf("\n %s is a valid operator\n",yytext);
            return '+';}
[-] {printf("\n %s is a valid operator\n",yytext);
            return '-';}

[/] {printf("\n %s is a valid operator\n",yytext);
            return '/';}
[*] {printf("\n %s is a valid operator\n",yytext);
            return '*';}

\n {return NL;}

. {return yytext[0];}
%%
```

Harsh (4016)

yacc1.y file

```
%{
        #include "yaac1.tab.h"
%}

%token NUM VAR NL

%%
        #include<stdio.h>
        #include<stdlib.h>


%left '+' '-' '*' '/' ;

S : S1 NL{printf("\nValid Expression\n");return 0;}

S1 :  S1 '+' S1|S1 '-' S1|S1 '/' S1|S1 '*' S1| '(' S1 ')' | VAR | NUM |;

%%

int main(){
        printf("\nEnter an Expression :: ");
        yyparse();
    return 0;
}

int yywrap(){}

int yyerror(){
        printf("\nInvalid Expression\n");
        exit(1);
}
```

Harsh (4016)

```
D:\Flex Windows\Bison\bin>bison -d yaac1.y

D:\Flex Windows\Bison\bin>flex yaac1.l

D:\Flex Windows\Bison\bin>gcc lex.yy.c yaac1.tab.c

D:\Flex Windows\Bison\bin>a.exe

Enter an Expression :: 5+6

 5 is a valid number

 + is a valid operator

 6 is a valid number

Valid Expression

D:\Flex Windows\Bison\bin>a.exe

Enter an Expression :: (5-7

 5 is a valid number

 - is a valid operator

 7 is a valid number

Invalid Expression

D:\Flex Windows\Bison\bin>
```

Harsh (4016)

Question -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 .

yacc2.l file

```
%{
  #include <stdio.h>
  #include <stdlib.h>
  #include "yacc2.tab.h"
%}

%option noyywrap
%%
[a-z] { return L; }
[0-9] { return D; }
[ \t\n]+ { ; }
. { return yytext[0]; }
%%
```

yaac2.y file

```
%{
  #include <stdio.h>
  #include <stdlib.h>
  #include "yacc2.tab.h"
%}

%token D L
%%
S : L D { printf("VALID IDENTIFIER\n"); }
  ;
%%
int main()
{
```

```
    printf("\n Enter identifier\n");
    yyparse();
    return 0;
}

int yywrap(){}

int yyerror(){
        printf("\nInvalid Identifier\n");
        exit(1);
}
```

```
D:\Flex Windows\Bison\bin>bison -d yacc2.y

D:\Flex Windows\Bison\bin>flex yacc2.l

D:\Flex Windows\Bison\bin>gcc lex.yy.c yacc2.tab.c

D:\Flex Windows\Bison\bin>a.exe

 Enter identifier
e3
VALID IDENTIFIER
3e

Invalid Identifier

D:\Flex Windows\Bison\bin>
```

Harsh (4016)

Question -1 A program in yacc to evaluate an expression (simple calculator program for addition and subtraction,mutiplication ,divison).

yaac3.l file

```
%{

        #include<stdio.h>
        #include<stdlib.h>
        #include "yaac3.tab.h"
        int yylval;
%}

%%

[0-9]+ {yylval = atoi(yytext);
       return NUM;}

[\t]+   ;

\n  {return 0;}

.   {return yytext[0];}

%%
```

yaac3.y file

Harsh (4016)

```
%{

        #include<stdio.h>
        #include<stdlib.h>
        #include "yaac3.tab.h"

%}

%token NUM
%left '+' '-'
%left '/' '*'
%left '(' ')'

%%
expr:e{printf("Result is :: %d\n",$$);
      return 0;}

e:e '+' e{$$ = $1+$3;}
 |e '-' e{$$ = $1-$3;}
 |e '*' e{$$ = $1*$3;}
 |e '/' e{

  if($3==0){
  printf("\nDivision By Zero\n");
  printf("Result is :: Undefined");
  return 0;

  }
  else
  $$ = $1/$3;}

 |'(' e ')'{$$ = $2;}
 |NUM {$$ = $1;}

%%

int main(){
```

```
    printf("\nEnter the arithmetic expression ::");

        yyparse();
        printf("\nValid Expression\n");

        return 0;

}

int yywrap(){
        return 0;
}

int yyerror(){
        printf("\nInvalid Expression\n");
        exit(1);
}
```

Harsh (4016)

```
D:\Flex Windows\Bison\bin>bison -d yaac3.y

D:\Flex Windows\Bison\bin>flex yaac3.l

D:\Flex Windows\Bison\bin>gcc lex.yy.c yaac3.tab.c

D:\Flex Windows\Bison\bin>a.exe

Enter the arithmetic expression ::5+6
Result is :: 11

Valid Expression

D:\Flex Windows\Bison\bin>a.exe

Enter the arithmetic expression ::6-1
Result is :: 5

Valid Expression

D:\Flex Windows\Bison\bin>a.exe

Enter the arithmetic expression ::2*3
Result is :: 6

Valid Expression

D:\Flex Windows\Bison\bin>a.exe

Enter the arithmetic expression ::4/2
Result is :: 2

Valid Expression

D:\Flex Windows\Bison\bin>a.exe

Enter the arithmetic expression ::4/0

Division By Zero
Result is :: Undefined
```

Harsh (4016)

Question -12 A program in yacc to recognize the strings “ab”,”aabb”,... of the language (anbn,n>=1).

yacc4.l file

```
%{
        #include<stdio.h>
        #include<stdlib.h>
        #include "yacc4.tab.h"


%}

%option noyywrap
%%
[a] { return A; }
[b] { return B; }
[ |\n|\t] { return yytext[0]; }
. { return yytext[0]; }
%%
```

yacc4.y file

```
%{

        #include<stdio.h>
        #include<stdlib.h>
        #include "yacc4.tab.h"
%}

%token A B

%%
S : E '\n' { printf("VALID STRING\n"); exit(0); }
  ;
```

Harsh (4016)

```
E : A E B
  | A B
  ;
%%

int main(){
        printf("\nEnter the string :: ");
        yyparse();
        return 0;
}

yywrap(){}
yyerror(){
        printf("\nInvalid String");
}
```

```
D:\Flex Windows\Bison\bin>bison -d yacc4.y

D:\Flex Windows\Bison\bin>flex yacc4.l

D:\Flex Windows\Bison\bin>gcc lex.yy.c yacc4.tab.c

D:\Flex Windows\Bison\bin>a.exe

Enter the string :: aabb
VALID STRING

D:\Flex Windows\Bison\bin>a.exe

Enter the string :: aaab

Invalid String
D:\Flex Windows\Bison\bin>
```

Harsh (4016)

Question -13 A program in yacc to recognize the language (anb ,n>-10).(output to say input is valid or not)

yaac5.l file

```
%{
        #include<stdio.h>
        #include<stdlib.h>
        #include "yaac5.tab.h"
%}

%%
[a] {return A;}
[b] {return B;}
\n  {return NL;}
. {return yytext[0];}
%%
```

yaac5.y file

```
%{
        #include<stdio.h>
        #include<stdlib.h>
        #include "yaac5.tab.h"

%}

%token A B NL

%%
S : A A A A A A A A A A S1 B NL
    { printf("\nValid String \n");
      return 0;}
```

Harsh (4016)

```
S1 : A S1
   |;

%%

main(){

        printf("\nEnter a String :: ");
        yyparse();

}

yywrap(){}

yyerror(){
        printf("\nInvalid String\n");
        return 0;
}
```

```
D:\Flex Windows\Bison\bin>bison -d yaac5.y

D:\Flex Windows\Bison\bin>flex yaac5.l

D:\Flex Windows\Bison\bin>gcc lex.yy.c yaac5.tab.c

D:\Flex Windows\Bison\bin>a.exe

Enter a String :: aaaaaaaaaaaab

Valid String

D:\Flex Windows\Bison\bin>a.exe

Enter a String :: aaaab

Invalid String

D:\Flex Windows\Bison\bin>
```