

Part B- Compiler Design Lab- Question Bank

1. Write a C / C++ program to accept a C program and do error detection & correction for the following.
(CO1)

Check for unterminated string constants in the input C program. i.e A string constant begins with double quotes and extends for more than one line. Intimate the error line numbers and the corrective actions to the user.

Ans:

```
#include<stdio.h>
#include<string.h>
int main()
{
    FILE *fp;
    int strcheck=0;
    int i;
    int lineno=0;
    int string=0;
    char line[100];
    int open,close;
    clrscr();
    fp=fopen("file.txt", "r");
    if(fp==NULL)
    {
        printf("File cant be opened\n"); exit(0);
    }
    printf("File opened correctly!\n");
    while(fgets(line, sizeof(line), fp)!=NULL)
    {
        lineno++;
        strcheck=0;
        string=0;
        open=close=0;
        for(i=0;i<strlen(line);i++)
        {
            if(line[i]=="")
            {
                string=1;
                if(open==1&&close==0)
                    close=1;
                else if(open==0&&close==0)
                    open=1;
                else if(open==1&&close==1)
                    close=0;
            }
        }
        if(open==1 &&close==0)
        {
```

```

        printf("\n Unterminated string in line %d. String Has to be closed",lineno); strcheck=1;
    }
    else if(string==1 && strcheck==0)
    {
        printf("\n String usage in line %d is validated!",lineno);
    }
}
}

```

file.txt :

```

#include <stdio.h>
#include <conio.h>
int s[35]="gh";
void main()
{ int a;
char c[10]="msrit",f[]="lk;
strlen("hijkl);
a=a+/*b;
}

```

2. Write a C / C++ program to accept a C program and do error detection & correction for the following.

(CO1)

Check for un- terminated multi line comment statements in your C program.

Ans:

```

#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int main()
{
    FILE *fp;
    int commentcheck=0;
    int i;
    int lineno=0;
    int comment=0;
    char line[100];
    int open=0,close=0,openlineno,closelineno;
    //clrscr();
    fp=fopen("file2.txt", "r");
    if(fp==NULL)
    {
        printf("File cant be opened\n"); exit(0);
    }
    printf("File opened correctly!\n");
    while(fgets(line, sizeof(line), fp)!=NULL)
    {
        lineno++;
        //getch();
        commentcheck=0;
        comment=0;
        if(open==1&&close==0)

```

```

        printf("\n%s",line);
if(strstr(line,"/*")&&open==0)
{
    open=1;close=0; comment=1;
    openlineno=lineno;
    printf("\n%s",line);
}
if(strstr(line,"*/")&&close==0&&open==1)
{
    closelineno=lineno;
    if(open==1&&close==0)
    {
        close=1;
        open=0;
        printf("\n Comment is displayed above!\nComment opened in line no%d
and closed in line no %d",openlineno,closelineno);
    }
}
}
if(open==1&&close==0)
{
    printf("\n  Unterminated  comment  in  begin  in  line  no  %d.  It  Has  to  be
closed",openlineno);
    commentcheck=1;
}
else if(comment==1 && commentcheck==0)
{
    printf("\n Comment usage in line %d is validated!",lineno);
}
return 0;
}

```

file.txt:

```

#include <stdio.h>
#include <conio.h>
/* char c[10]="msrit",f[]="lk; */
strlen("hijkl");
/*dgdfgdfg*/
a=a+b;
/* fsdgdgds sdgfsd sdfsd
*/

```

3. Write a Lex program to accept a C program and do error detection & correction for the following.
(CO1)

Check for un-terminated string constants in the input C program. i.e A string constant begins with double quotes and extends for more than one line. Intimate the error line numbers and the corrective actions to the user

Ans:

```

%{
#include<stdio.h>

int c=0;
FILE *fp;
%}

%%

\n { c++; }

["][a-zA-Z0-9]*["] {ECHO; printf(" Valid String in line number %d\n ",c+1);}

["][a-zA-Z0-9]* { ECHO; printf(" InValid String in line number %d\n ",c+1);}

. ;

%%

void main()
{
yyin=fopen("source.txt","r");
yylex();
fclose(yyin);
}

file.txt :

#include <stdio.h>
#include <conio.h>
void main()
{
int a,b,h;
a=a+b;
char d[20]="d",h[67]="yu ;
char c[10]="msrit";
a=a+/b+h;
strlen("msrit");
strlen("msr");
strcpy(c,"Bangalore");
b=b+*; }

```

4. Write a Lex program to accept a C program and do error detection & correction for the following.(CO1)

Check for valid arithmetic expressions in the input C program. Report the errors in the statements to the user.

Ans:

```
%{
#include<stdio.h>
int c=0; FILE *fp;
%}

operator [-+*/]
identifier [a-zA-Z][a-zA-Z0-9-]*
number [0-9]+
expression ({identifier}|{number}){operator}({identifier}|{number})
%%

\n { c++; }
^"#" .+ ;
^("int "|"float "|"char ").+ ;
"void main()" ;
{identifier}"="({expression}+";") { printf("Valid expression in line no :%d\t",c+1);ECHO;printf("\n");}
{identifier}"="({number}|{identifier}";") { printf("Valid expression in line no :%d\t",c+1);ECHO;printf("\n");}
({number}|([0-9]*[a-zA-Z0-9-]+))"=" {expression}+ { printf("Invalid expression in line no : %d;Lvalue should satisfy the identifier rules\n",c+1);ECHO;printf("\n");}
{identifier}"="; { printf("Invalid expression in line no : %d; R-value required; Expression is needed at right hand side of assignment operation\n",c+1);ECHO;printf("\n");}
{operator}{operator}+ {printf(" Invalid expression in line no: %d;More than one operator can't be used in expression consequetively",c+1);ECHO;printf("\n");}
.\n ;
%%

void main()
{
yyin=fopen("source.txt","r");
yylex();
fclose(yyin);
}

file.txt:
#include<stdio.h>
#include <conio.h>
#include <stdlib.h>
void main()
{
int a=1s,b,h;
a=a+b;
a=a+/b+h;
1a=7+j-;
a=;
b=b+*; }
```

5. Write a Lex program to accept a C program and do the following error detection & correction.(CO1)

Check for the valid usages of numerical constants in the input C program. Intimate the invalid usages to user.

Ans:

```
%{  
  
#include<stdio.h>  
  
int c=0;  
  
%}  
  
number [0-9]+(".")?[0-9]*  
  
invalid [0-9]+(".")[0-9]*(((".")[0-9]*)+  
  
%%  
  
\n {c++;}  
  
{number} {printf("\nValid number in line number %d : ",c+1);ECHO;printf("\n");}  
  
{number}[a-zA-Z0-9_]+ {printf("\nInvalid number in line number %d: Number followed with alphabets  
is invalid",c+1);ECHO;printf("\n");}  
  
{invalid} {printf("\nInvalid number in line number %d: Number with more than one decimal point is  
invalid",c+1);ECHO;printf("\n");}  
  
. ;  
  
%%  
  
void main()  
  
{  
  
yyin = fopen("source.txt","r");  
  
yylex();  
  
fclose(yyin);  
  
}  
  
file.txt:  
  
#include <stdio.h>  
#include<stdlib.h>  
void main()  
{  
int a=56;  
a=1b;  
a=a+5h;  
a=a+4.5+5. 6.6; }
```

6 Check for valid declarative statements in your program. (CO1) eg: int a,b;

Ans:

```
%{
#include<stdio.h>
int c=0;
%}
%s DECLARE VAR
identifier [a-zA-Z][a-zA-Z0-9-]*
number [0-9]+[.]?[0-9]*
string ("")(a-zA-Z0-9+)(")
%%
\n {c++;}
"int " | "float " {BEGIN DECLARE;}
<DECLARE>{identifier}{"="{number}}? {BEGIN VAR;}
<DECLARE>{identifier}{"="{string}} {BEGIN VAR; printf("\n Invalid variable declaration in line no
%d;string can't be assigned to integer or float variable:",c+1);ECHO;printf("\n");}
<VAR>";" {BEGIN 0;}
<VAR>{identifier}{"="{number}}? {}
<VAR>{identifier}{"="{string}} {printf("\n Invalid variable declaration in line no %d; string can't be
assigned to integer or float variable:",c+1);ECHO;printf("\n");}
<VAR>\n {BEGIN 0; c++;}
<VAR>"," {BEGIN DECLARE;}
<VAR>[,][,]+ {printf("\n Invalid usage of more than one comma in declaration in line no
%d",c+1);BEGIN DECLARE;ECHO;printf("\n");}
.;
%%
void main()
{
yyin = fopen("source.txt","r");
yylex();
fclose(yyin);
}
```

file.txt

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
void main()
{ int a,b=78,g="78",,,;
float c=5.6,h="fg";
sa=5;
a=a+b;
printf("\n "); }
```

7. Write a Lex program to accept a C program and do the following error detection & correction.(CO1)
Check for the valid if statement in the input C program. Report the errors to users.

Ans:

```
%{
#include<stdio.h>
int c=0,bc=0,fc=0;
FILE *fp;
}%
%s IF OPENP CLOSEP OPENF
%%
\n { c++; }
"if" {BEGIN IF;ECHO;bc=0;}
<IF>\n {c++;ECHO;printf("\n");}
<IF>"(" {BEGIN OPENP;ECHO;bc++;}
<IF>")" {BEGIN CLOSEP;ECHO;bc--;}
<OPENP>")" {ECHO;bc--;BEGIN CLOSEP;}
<OPENP>"(" {ECHO;bc++;}
<OPENP>. {ECHO;}
<CLOSEP>"{" {if(bc==0) {printf("condn is valid in line no %d\n",c+1);} else printf("condn invalid in
line no %d;Paranthesis mismatch in condn\n",c+1);BEGIN OPENF;ECHO;printf("\n");fc++;}
<CLOSEP>"(" {BEGIN OPENP;bc++;ECHO;}
<CLOSEP>")" {ECHO;bc--;}
<CLOSEP>. {ECHO;}
<CLOSEP>\n {ECHO;printf("\n");c++;}
<OPENF>")" {fc--;if(fc==0) BEGIN 0;;ECHO;printf("\n");}
<OPENF>. {ECHO;}
<OPENF>\n {ECHO;c++;}
.| \n ;
%%
void main()
{
yyin=fopen("source.txt","r");
yylex();
fclose(yyin);
}
```

file.txt

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main() {
int a,b=78;
if((a<5&&j<9) {
a=a+h;
```



```

g=6+7;
a=a+b;
printf("\n");
}
if(a<n)
{ h=j+k;
}
if(a<n))
{ g=h+k;
}
}

```

8. Check for un- terminated multi line comment statement in your C program.(CO1)

Ans:

```

%{
#include<stdio.h>
int c=0,oc=0;
FILE *fp;
%}
%s COMMENT
%%
\n {c++;}
"/*" {BEGIN COMMENT;printf("\n comment begins in line no : %d\n",c);ECHO;oc=1;}
<COMMENT>"*/" {BEGIN 0;ECHO;oc=0;printf(": Comment ends in line no %d\n",c);}
<COMMENT>\n {c++;printf("\n");ECHO;}
<COMMENT>. {ECHO;}
.;
%%
void main()
{
yyin=fopen("source.txt","r");
yylex();
fclose(yyin);
if(oc==1)
{
printf("\n comment is not closed till the end of file!");
}
}

```

file.txt:

```

#include<stdio.h>
#include<conio.h>
#include<string.h>
/*dfddf*/
void main()
{
/*vbhfgfhfgh
dfhfgh

```

```

fghgfhfg
fghfh */
int a,b=78;
if((a<5&& j <9)
{ a=a+h;
g=6+7;
a=a+b;
printf("\n");
}
/*if(a<n) {
h=j+k; }
if(a<n))
{ g=h+k;
}
}

```

9. Write Yacc program to accept a statement and do the following error detection.(CO2)

a) Check for valid arithmetic expressions in the input C statement. Evaluate the arithmetic expression.

Ans:

```

%{
#include<stdio.h>
int flag=1; %}
%token id num
%left '(' ')'
%left '+' '-'
%left '/' '*'
%nonassoc UMINUS
%%

stmt: expression { printf("\n valid
exprn");}
;

expression : '(' expression ')' | '(' expression {printf("\n Syntax error: Missing right paranthesis");}
| expression '+' expression {printf("\n plus recog!"); $$=$1+$3; printf("\n %d", $ $);}
| expression '+' { printf ("\n Syntax error: Right operand is missing ");}
| expression '-' expression {printf("\n minus recog!"); $$=$1-$3; printf("\n %d", $ $);}
| expression '-' { printf ("\n Syntax error: Right operand is missing ");}
| expression '*' expression {printf("\n Mul recog!"); $$=$1*$3; printf("\n %d", $ $);}
| expression '*' { printf ("\n Syntax error: Right operand is missing ");}
| expression '/' expression {printf("\n division recog!"); if($3==0) printf("\n division cant be done, as
divisor is
zero.");}
else {$$=$1+$3; printf("\n %d", $$);}
| expression '/' { printf ("\n Syntax error: Right operand is missing ");} | expression '%' expression |
expression '%' { printf ("\n Syntax error: Right operand is missing ");}

```

```

| id
| num
; %% main() { printf(" Enter an arithmetic expression\n");
yyparse(); } yyerror() { printf(" Invalid arithmetic
Expression\n"); exit(1);
}
Lex:
%{
#include "y.tab.h"
#include<stdio.h>
#include<ctype.h
> extern int
yylval; int val;
%}
%%
[a-zA-Z][a-zA-Z0-9]* {printf("\n enter the value of variable
%s:",yytext);scanf("%d",&val);yylval=val;return id;}
[0-9]+[.]?[0-9]* {yylval=atoi(yytext);return num;}
[ \t] ;
\n {return 0;}
. {return yytext[0];}
%% int yywrap()
{ return
1;
}

```

