



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

PAGE NO.

Write a YACC program to recognize string $a^n b^n$, where $b \geq 0$

Lex file

```
%  
#include "y.tab.h"  
%  
%%  
a      return 'a' ;  
b      return 'b' ;  
[ ]    return 0 ;  
return yyltext[0] ; //catch-all unrecognized  
                           characters.  
%%
```

YACC file

```
%  
#include <stdio.h>  
#include <stdlib.h>  
int yylval ;  
int yyleng ;  
int yyl;
```

```
void error(void "const char");
```

```
void error(const char *s);
```

```
% %
```

```
% token a, b
```

```
% %
```

```
start:
```

```
sequence '\n' { printf("Valid String: Matches
```

```
a^n b^n \n"); }
```

```
'\n'
```

```
{ printf("empty \n"); }
```

```
sequence:
```

```
'a' sequence 'b' { /* ensure matching a^n b^n */ }
```

```
1 /* empty */
```

```
;
```

```
% %
```

```
void yyerror(const char *s) {
```

```
    fprintf(stderr, "Error: %s\n", s);
```

```
}
```

```
int main()
```

```
{ printf("Enter a string (a^n b^n, n >= 0): ");
```

```
    yyparse();
```

```
    return 0;
```

```
}
```




POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

PAGE NO. _____

Write a YAcc program to evaluate an arithmetic expression involving operators +, -, *, /.

Lex program

% §

/* definition section */

#include <stdio.h>

#include <stdlib.h>

#include "y.tab.h"

extern int yylval;

% }

/* rules section */

%%

[0-9] + § yylval = atoi(yytext);

· return Number;

}

[/ t] ;

[/ n] return 0;

· return yytext[0];

%%

int yywrap()

§ return 1;

}

parser source code

```
% S
```

```
/* definition section */
```

```
#include <stdio.h>
```

```
int flag = 0;
```

```
%}
```

```
% token Number
```

```
% left '+' '-'
```

```
% left '*' '/' '%'
```

```
% left '(' ')' 
```

```
/* rules section */
```

```
%%
```

```
ArithmeticExpression : E {
```

```
    printf (" \n Result = %d \n", $1);
```

```
    return 0;
```

```
};
```

```
E : E '+' E { $1 = $1 + $3; }
```

```
| E '-' E { $1 = $1 - $3; }
```

```
| E '*' E { $1 = $1 * $3; }
```

```
| E '/' E { $1 = $1 / $3; }
```

```
| E '%' E { $1 = $1 % $3; }
```

```
| '(' E ')' { $1 = $2; }
```

```
| Number { $1 = $1; }
```

```
% %
```




POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

PAGE NO.

// driver code

void main()

{
printf("\n enter any arithmetic expression
which have +, -, /, *, %, \n");

yyparse();

if (flag == 0)

printf("enter expression is valid \n");

}

void yyerror()

{
printf("Invalid expression \n");

flag = 1;

}



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

PAGE NO.

Write a YACC program to check validity of a strings $abcd$, $aabbbcd$ using grammar

$a^n b^m c^1 d^1 m$, where $n, m > 0$

% ξ Lexical Analyzer program

/* definition section */

#include "y.tab.h"

% $\{$

/* rule section */

%%

$[aA]$ ξ return A; $\}$

$[bB]$ ξ return B; $\}$

$\backslash n$ ξ return NL; $\}$

ξ return yytext[0]; $\}$

%%

int yywrap()

ξ return 1;

$\}$


```
/* parser source code + /
```

```
%$ % definition section +/
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
% 3
```

```
% token A B NL
```

```
/* Rule section + /
```

```
% %
```

```
$tmt : A A A A B NL { printf ("valid string\n");
```

```
exit(0); }
```

```
;
```

```
S : SA
```

```
|
```

```
;
```

```
% %
```

```
int yyerror(char *msg)
```

```
{ printf ("Invalid string\n");
```

```
exit(0);
```

```
}
```

```
// driver code
```

```
main()
```

```
{ printf ("enter the string\n");
```

```
yyparse();
```

```
}
```



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

PAGE NO.

Write a C program to find first of any grammar.

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#define MAX 10

void findFirst(char, int, int);
int n;
char production[MAX][MAX], first[MAX];
int main()
{
    int i;
    char choice, c;
    printf("Enter the no. of productions: ");
    scanf("%d", &n);
    printf("Enter the production eg. E = E + T or T = a\n");
    for(i = 0; i < n; i++)
        scanf("%s", production[i]);
}
```



```

do
{
    printf("enter the non-terminal to find first: ");
    scanf("%c", &c);
    int start = 0;
    findFirst(c, 0, start);
    printf("FIRST(%c) = {", c);
    for (i = 0; i < start; i++)
        printf("%c", first[i]);
    printf("} \n");

    printf("do you want to continue?(y/n): ");
    scanf("%c", &choice);
}
while (choice == 'y' || choice == 'Y');

return 0;
}

void findFirst(char c, int q1, int q2)
{
    int j;
    if (!isupper(c)) {
        first[q2++] = c; // Terminal directly goes to first
        return;
    }
    for (j = 0; j < n; j++) {
        if (production[j][0] == c) {
            if (islower(production[j][2]) || production[j][2] == '#')
                first[q2++] = production[j][2];
            else findFirst(production[j][2], q1, q2);
        }
    }
}

```