CD assignment K Jeevan Sai

set-3 AP18110010266

1) Given Expression

$x = y + z \times 15$

$x = y + z \times 15$

↓

Lexical analyser

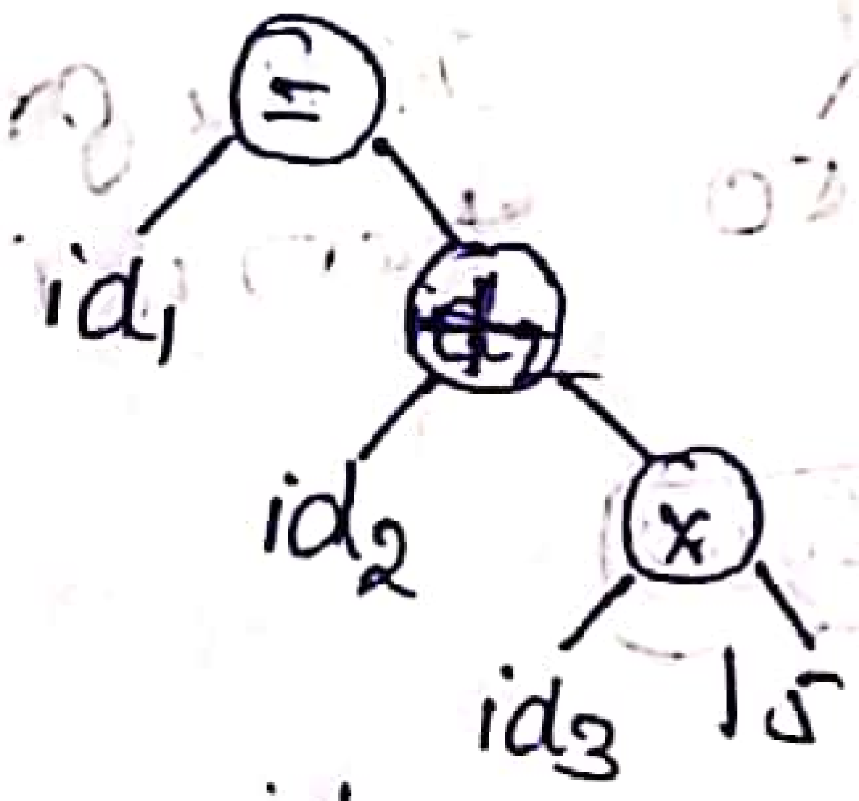
↓

$id_1 = id_2 + id_3 \times 15$

↓

Syntax analyser

↓

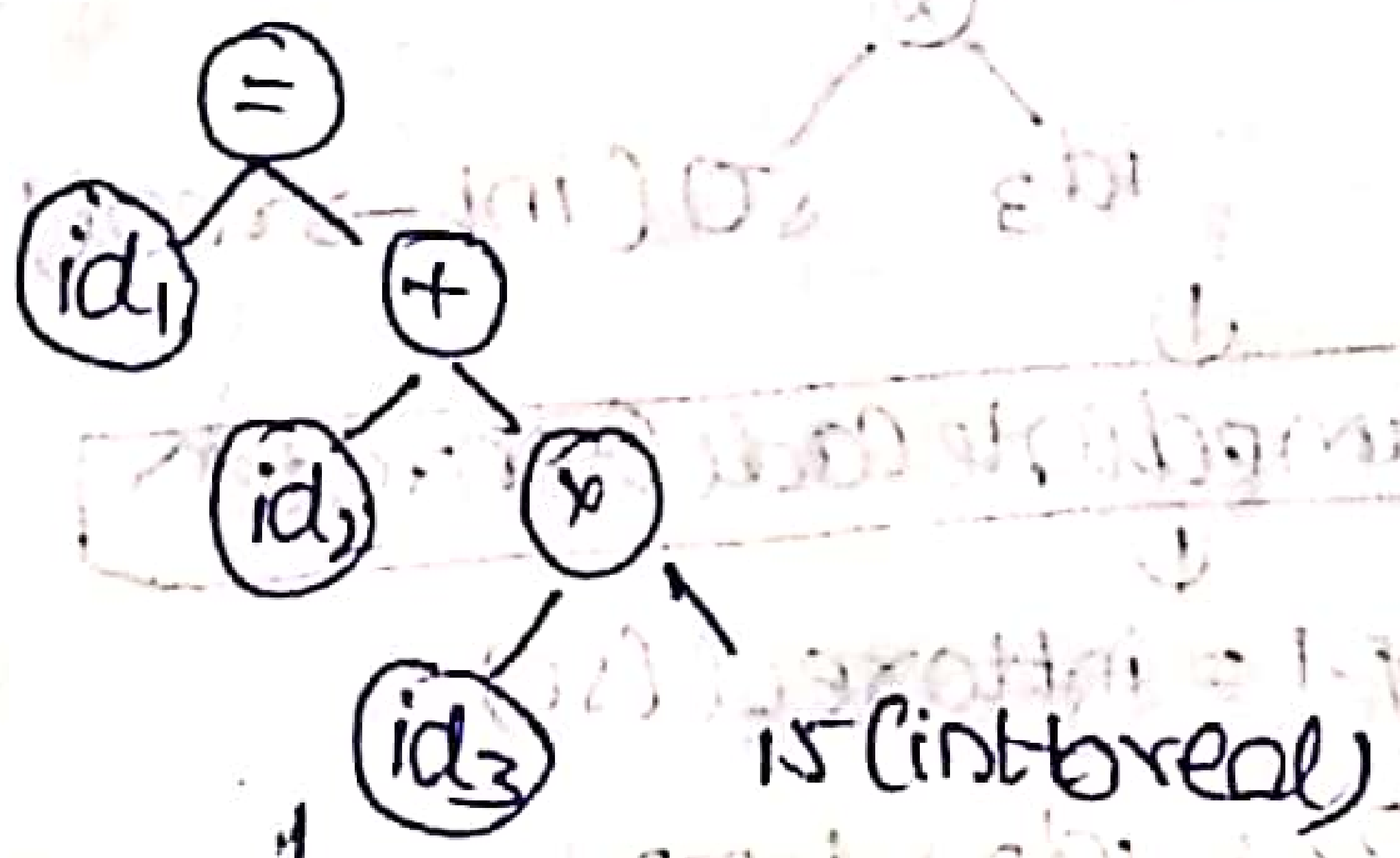


This parse tree is used to check weather the syntax of the expression is correct or not

↓

Semantic analyser

↓



↓

Intermediate code Generator

↓

temp1 = inttoreal(15)

temp2 = id3 * temp1

temp3 = id2 + temp2

~~temp4~~ = id1 = temp3

↓

Code optimization

↓

temp1 = id3 * 15.0

id1 = id2 + temp1 →

Code Generator

↓

MOVF R2, id3

MULF R2, #15.0

MOVF R1, id2

ADDF R1, R2

MOVF id1, R1

2) Define

a) lexeme: AP18110010266

sequence of charecters grouped as one unit

b) token: sequence of charicters that can be generated lexeme

token can be identifier, keywords, operator and punctuation marks.

(c) pattern:

It is a form that each diff lexeme takes

```
 1         2       3  4  5 6 7  8 9
void  multiply(int i, int j)
10
{      11   12  13
    int temp;
    14   15 16 17 18 19
    temp=i*j;
    20      21  22
    return temp;
}
23
```

23 Tokens are present

Lexeme	Tokens	Patterns
void	keyword	v→o→i→d
multiply	identifier	letter(letter/digit)*
(	delimeter	,/;/,/"/'/{/}/!/\/[/]/(/)
int	keywort	i→n→t
i	identifier	letter(letter/digit)*
,	delimiter	,/;/,/"/'/{/}/!/\/[/]/(/)
j	identifier	letter(letter/digit)*
)	delimiter	,/;/,/"/'/{/}/!/\/[/]/(/)
{	delimiter	
temp	identifier	letter(letter/digit)*
;	delimiter	,/;/,/"/'/{/}/!/\/[/]/(/)
=	operator(rel)	</<=/=/>/>=/<>/
*	operator(Arth)	+/-/*//%/*
return	keyword	r→e→t→u→r→n
~~temp~~ }	delimiter	,/;/,/"/'/{/}/!/\/[/]/(/)

↳ repeated once are removed

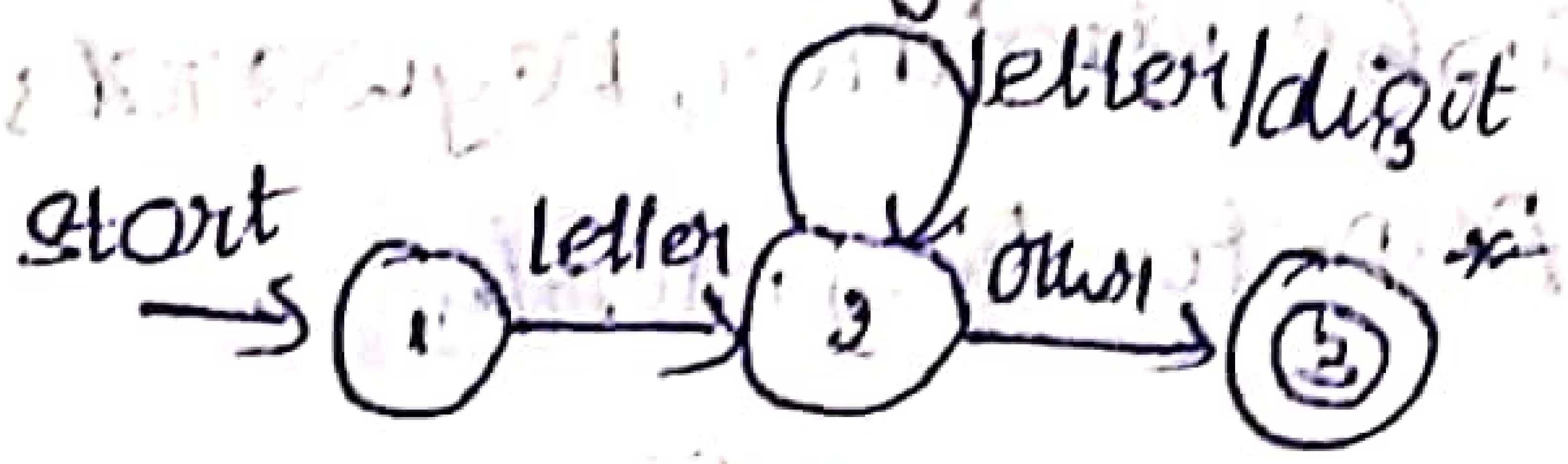
AP18110010266

3) regular Expressions and transtition diagram

(a) Identifier   digit => [0-9]. letter [a-zA-Z]

regex => letter (letter | digit)*

Transtition diagram:-



return(gettoken(), install_id())

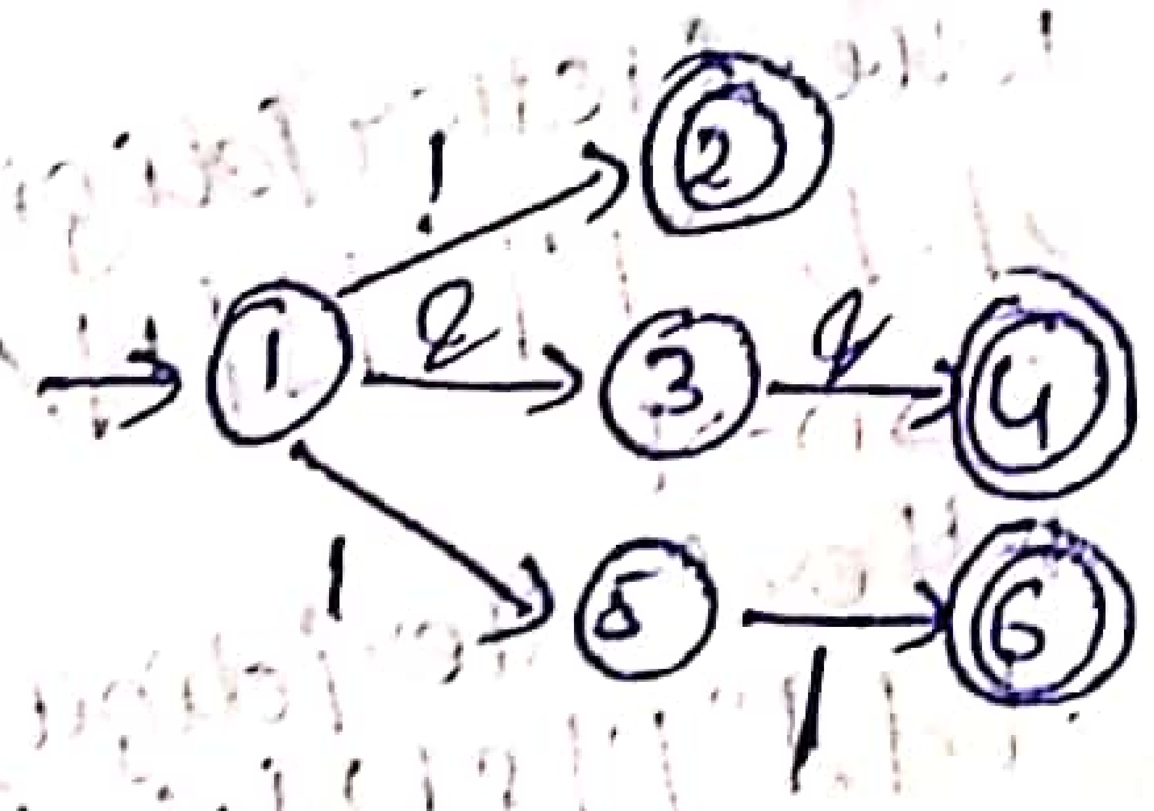
(b) Logical operators

regex: AND|OR|and|or|XOR|&&|<=|<|>|>=|!=|==

(In general) |&|OR|

(In C) &&| || | !

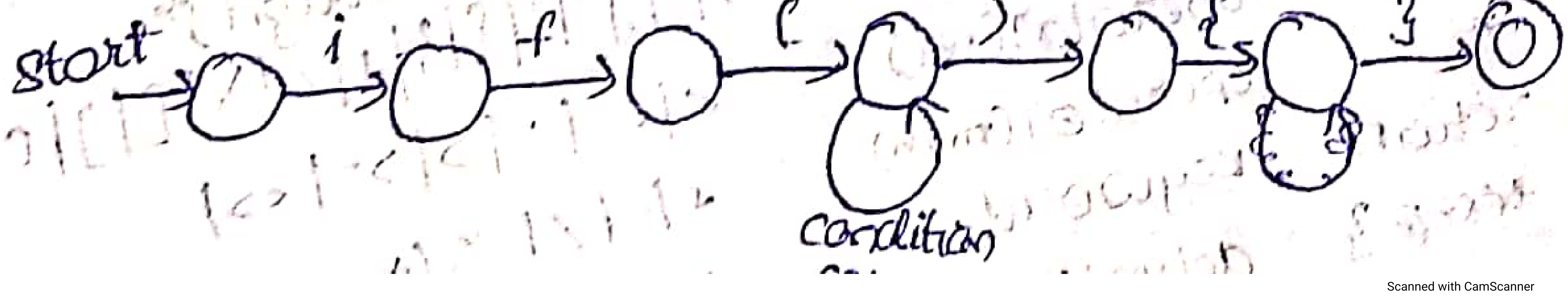
Transtition diagram



(c) If statement

Regex For simple if statement (?(-A)X)

Transtition diagram for if:



AP18110010266

4) a) Lex program to recognize identifier keyword and number)

```
%{
#include<stdio.h>
%}
%%
if|else|while|int|switch|for|char {printf("keyword");}
[a-z]([a-z]|[0-9])* {printf("identifier");}
[0-9]* {printf("number");}
.* {printf("invalid");}
%%
main()
{
  yylex();
  return 0;
}
int yywrap()
{
}
```

(or)

Store in symbol table

```
%{
#includ <stdio.h>
#include "ytab.h";
extern char *yyval;
int n=0;
%}
%%
"int" {n++; return INT;}
"char" {n++; return CHAR;}
[a-zA-Z_][a-zA-Z0-9]*{yyval=yytext; if (n==0){return ID;}
                                                return 0;}
%%
main()
{ yylex()
}
```

Balanced paranthesis

6)
```
%{
#include<stdio.h>
int i=0;
%}

open "{"
close "}"
%%
{open} {i++;}
{close} {i--;}
if (i<0)
 {
   printf("Parenthesis Match");
 }
};
%%
main(char* args[], int argv)
{
   yyin=fopen(args[1],"r+");
   yylex();
   if (i==0)
   {
     printf("All paranthes are matched");
   }
}
int yywrap(void)
{ return 1;
}
```

c)

(c) Remove comments

```
%{
%}

start \/\*
end \*\/
/* For Multi Line comment */
%%

 \/\/(.*); /* single line commant */
 {start}.*{end} /* Multiline commenr */
%%
int main(int k, char **argv)
{
    yyin=fopen("input.txt","r");
    yyout=fopen("output.c","w");
    yylex();
    return 0;
 }
```

(d) To add line number Before each Line in afile

```
 %{
    int line_number=1;
 %}
 line .*\n
 %%
 {line}{printf("%10d %s", line_number++, yytext);}
%%
int yywrap(){}
int main(~~int argc~~char *argv[])
{ extern File *yyin;
   yyin=fopen=("input.c", "r");
   yylex();
   return 0;
}
```