Date
Page

20181CSE0621
Sai Ram.K
7-CSE-10

Part-B

Q.1]

```
fact = 1;
db = 0;
for(i = 1; i <= 10; i++)
{
    if (fact == 120)
        break;
    fact = fact * i;
    db = db + fact;
}
```

a] Three Address Code

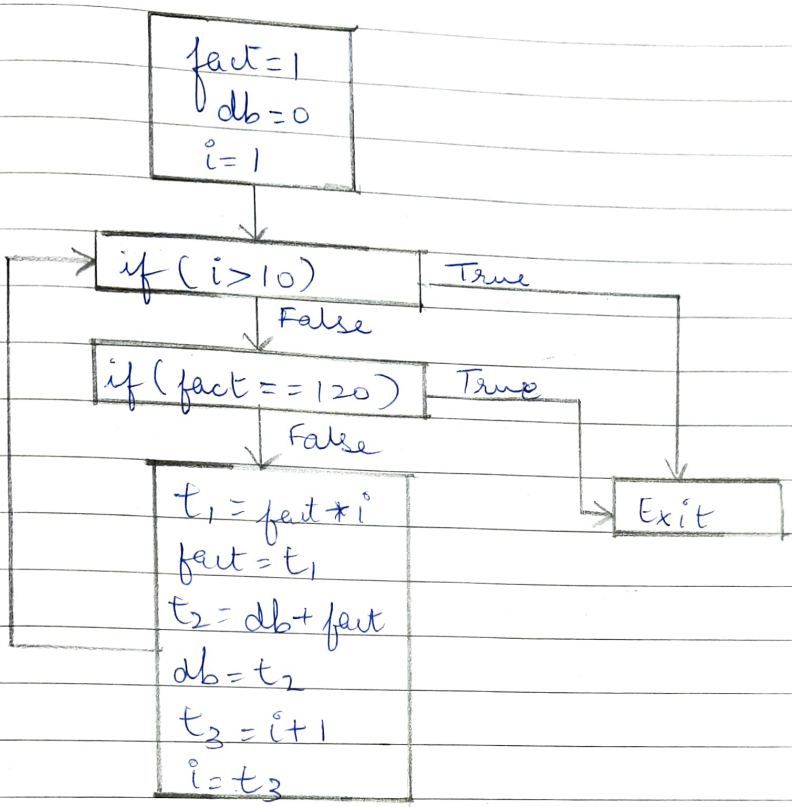
(1) fact = 1 ——> leader
(2) db = 0
(3) i = 1
(4) if (i > 10) goto 13 ——> leader
(5) if (fact == 120) goto 13 ——> leader
(6) t_1 = fact * i ——> leader
(7) fact = t_1
(8) t_2 = db + fact
(9) db = t_2
(10) t_3 = i + 1
(11) i = t_3
(12) goto 4
(13) exit

b] Leaders : (1) fact = 1
(4) if (i > 10) goto 13
(5) if (fact == 120) goto 13
(6) t_1 = fact * i

2018ICSE062
Sai Ram

classmate
Date
Page

Control flow graph:



20181CSE0621
Sai Ram

20181CSE0621
Sai Ram
7-CSE-10

Part-B

Q.2) E -> if(BA)
A -> andBA / orBA / ε
B -> true
B -> false

-> Augumented set

I_0: E' -> . E
E -> .if(BA)

I_1: Goto(I_0, E)
E' -> E.
Goto(I_0, if)

I_2: E -> if .(BA)
Goto(I_2, ()

I_3: E -> if (. BA)
B -> . true
B -> . false
Goto(I_3, B)

I_4: E -> if(B.A)
A -> . and BA
A -> . or BA
Goto(I_3, true)

I_5: B -> true .
Goto(I_3, false)

I_6: B -> false.
Goto(I_4, A)

I_7: E -> if(BA.)
Goto(I_4, and)

I_8: A -> and . BA
B -> . true
B -> . false
Goto(I_4, or)

I_9: A -> or . BA
B -> . true
B -> . false
Goto(I_7,))

I_{10}: E -> if(BA).
Goto(I_8, B)

I_{11}: A -> and B.A
A -> . and BA
A -> . or BA
I_5: Goto(I_8, true)
I_6: Goto(I_8, false)
Goto(I_9, B)

I_{12}: A -> or B.A
A -> . and BA
A -> . or BA
I_5: Goto(I_9, true)
I_6: Goto(I_9, false)
Goto(I_{11}, A)

I_{13}: A -> And BA.
Goto(I_{12}, A)

I_{14}: A -> or BA.

20181CSE0621
Sai Ram

Follow set : follow(E) = {$}
follow(A) = {) }
follow(B) = { and, or,) }

R_1 = E → if(BA)
R_2 = A → and BA
R_3 = A → OR BA
R_4 = B → true
R_5 = B → false

State	Action								Goto		
	if	and	or	true	false	(	)	$	E	A	B
0	S2								1		
1								Accept			
2						S3					
3				S5	S6						4
4		S8	S9								
5		R4	R4				R4				
6		R5	R5				R5				
7							S10				
8				S5	S6						11
9				S5	S6						12
10								R1			
11										13	
12										14	
13							R2				
14							R3				

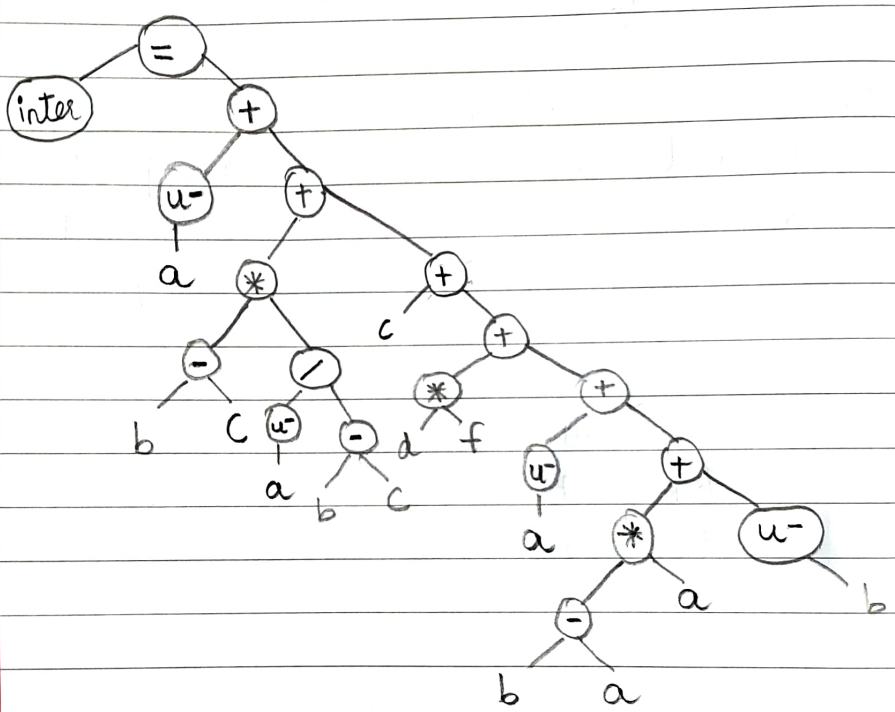
∴ The Grammar is in SLR(1).

2018ICSE0621
Sai Ram. K
7-CSE-10

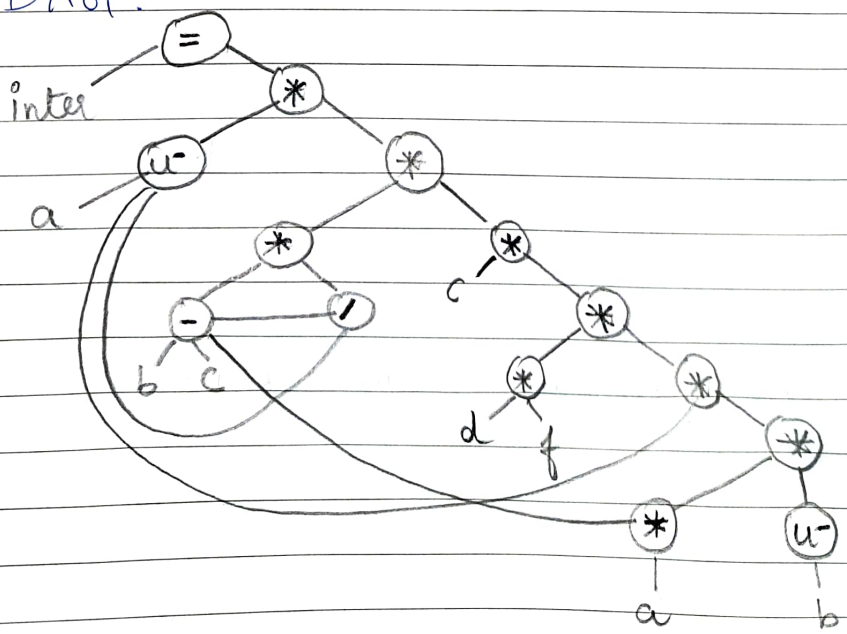
Part-B

Q.3] Given, inter = -a + (b-c) * -a / (b-c) + c + d * f + -a + (b-c) * a + -b

(i) Syntax tree: U- => UnaryMinus.



(ii) DAG:



20181CSE0624
Sai Ram

classmate
Date
Page

iii) Quadruples

	Operation	Arg1	Arg2	result
(0)	−	b	c	t_1
(1)	*	t_1	a	t_2
(2)	Uminus	b		t_3
(3)	+	t_2	t_3	t_4
(4)	Uminus	a		t_5
(5)	+	t_4	t_5	t_6
(6)	*	d		t_7
(7)	+	t_6	t_7	t_8
(8)	+	t_8	c	t_9
(9)	/	t_5	t_1	t_{10}
(10)	*	t_1	t_{10}	t_{11}
(11)	+	t_2	t_{11}	t_{12}
(12)	=	t_{12}		inta

iv) Tuples

	Operation	Arg1	Arg2
(0)	−	b	c
(1)	*	(0)	a
(2)	u^-	b	
(3)	+	(1)	(2)
(4)	u^-	a	
(5)	+	(3)	(4)
(6)	*	d	f
(7)	+	(5)	(6)
(8)	+	(7)	c
(9)	/	(4)	(0)
(10)	*	(0)	(9)
(11)	+	(1)	(10)
(12)	=	(11)	inta

20181CSE0621
Sai Ram
7-CSE-10

Part-B

Q.4] • A lexical analyzer by name yylex() must be provided.
• Using lex to produce yylex() is a common choice.
• It produces tokens consisting of a token name and its associated attribute value.

Q.5]

```
%{
#include <ctype.h>
#include <stdio.h>
#define YYSTYPE double
%}
%token NUMBER
%left '+' '-'
%left '*' '/'
%right UMINUS
%%
lines : lines expr '\n' {printf("%g\n", $2);}
      | lines '\n'
      | /* empty */
      ;
expr : expr '+' expr   { $$ = $1 + $3; }
     | expr '-' expr   { $$ = $1 - $3; }
     | expr '*' expr   { $$ = $1 * $3; }
     | expr '/' expr   { $$ = $1 / $3; }
     | '(' expr ')'    { $$ = $2; }
     | '-' expr %prec UMINUS { $$ = -$2; }
     | NUMBER
     ;
%%
```

2018ICSE0621
Sai Ram

classmate
Date
Page

```
yylex() {
    int c;
a] while ((c=getchar()) == ' ');
  if ((c == '.' || isdigit(c)))
  {
    ungetc(c, stdin);
    scanf("%lf", &yylval);
    return NUMBER;
  }
  return c;
}
```

b]
~~for ((c=0; c!=' ';)~~
```
for (c=0; c.getchar() != ' '; c++)
{
    ungetc(c, stdin);
    scanf("%lf", &yylval);
    return NUMBER;
}
    return c;
}
```

c]
```
if (c.getchar() != ' ')
{
    ungetc(c, stdin);
    scanf("%lf", &yylval);
    return NUMBER;
}
    return c;
}
```

20181CSE0621
Sai Ram

20181CSE0621
Sai Ram. K
7- CSE-10

Part - C

Q.1] Given, L -> WID — (1)
W -> while (c) begin S end — (2)
D -> do begin(S) end while(c) — (3)
C -> V <=> V — (4)
V -> id | num — (5)
S -> A S | ε — (6)
A -> ~~E~~ A + = V ; A | ; ~~A~~ — (7)

i] Tokens in the given grammar string are :-

L> Keywords : while, end, begin, do, id, num
L> Tokens : (If any) { , } , (,) , ; , : , >
L> Tokens : + , = , ; , <.

ii] Compute first :
first (L) = { while, do }
first (W) = ~~{ do }~~ { while }
first (D) = { do }
first (C) = { id, num }
first (S) = { id, num, ε }
first (A) = { id, num, ; }

Compute follow :
follow (L) = { $ }
follow (W) = { $ }
follow (D) = { $ }
follow (C) = {) }
follow (V) = { <, >, +, ; }
follow (S) = { end }
follow (A) = { id, ~~a~~ num, ;, end }

2018ICSE0621
Sai Ram

classmate
Date
Page

Parsing Table

	while
L	L → W
W	W → while (c) begin S end
D	
C	
V	
S	
A	

	while	(c)	Begin	End	do	<=	id	num	ε	+	;	$
L	L → W (1)				L → D (1)							
W	W → While (c) begin S end (2)											
D					D → do begin S end while (3)							
C							(4)	(4)				
V							(5)	(5)				
S							(6)	(6)	(6)	(6)		
A							(7)	(7)			(7)	

∴ The grammar is in LL(1).

2018ICSE0621
Sai Ram

iii] Input String validation