Name: Kumawat Lakhan Makhanlal
Roll No: 1906055
Branch: CSE-1
Course: Compiler Design

Course Code: CS 5404
Exam Date: 17/12/21

Solution 1)

$S \rightarrow L=R$ ----(1)

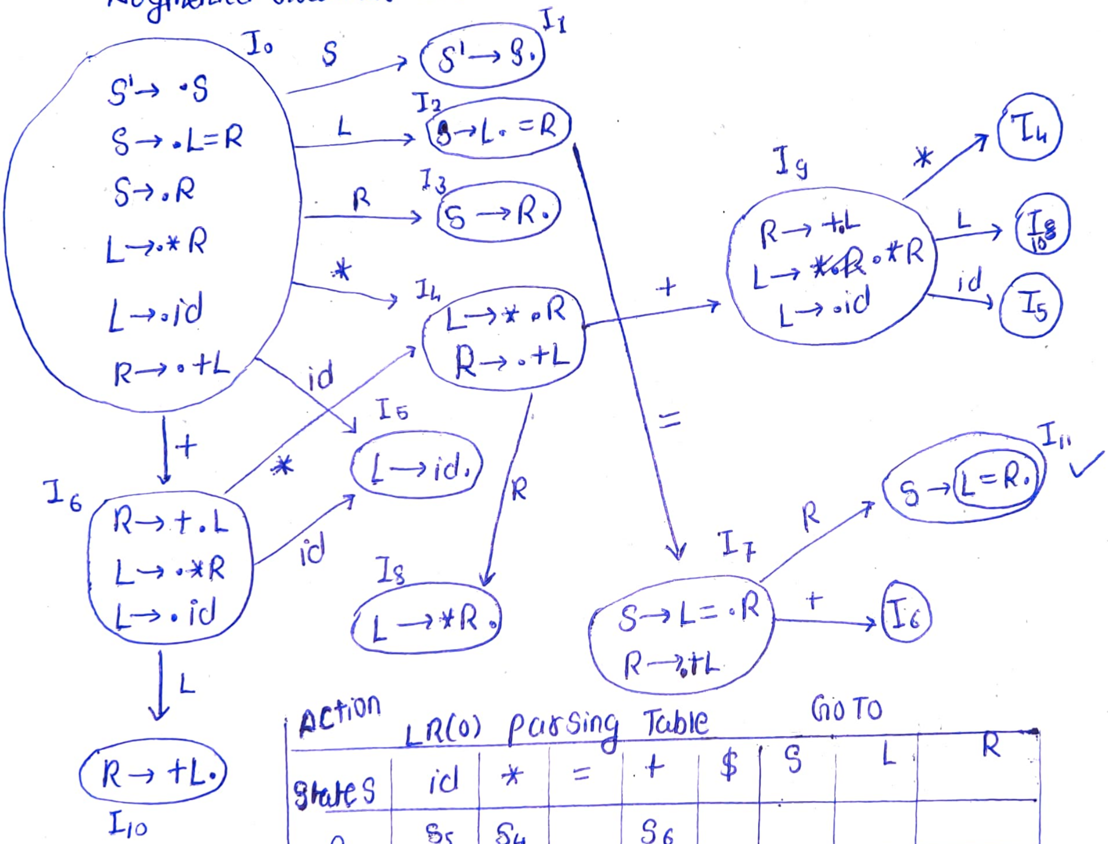
$S \rightarrow R$ ---(2)

$L \rightarrow *R$ --(3)

$L \rightarrow id$ ---(4)

$R \rightarrow +L$ ----(5)

Augmented Grammar and LR(0) items.



Action LR(0) parsing Table Go To

States	id	*	=	+	$	S	L	R
0	S5	S4		S6				
1				accept				
2			S7					
3	r2	r2	r2	r2	r2			
4								8
5	r4	r4	r4	r4	r4			
6	S5	S4					10	
7								11
8	r3	r3	r3	r3	r3			
9	S5	S4					10	
10	r5	r5	r5	r5	r5			
11	r1	r1	r1	r1	r1			

Name: Kumawat Lakhan Makhanlal
Roll No: 1906055
Branch: CSE-1
Course: Compiler Design

Course Code: CS5404
Exam Date: 17/12/2021

Solution 02)

Symbol Table : It is a Compile time data structure that is used by the Compiler to Collect and use information about the source program Constructs, such as variables, Constants, functions etc. The symbol table helps the compiler in determining and verifying the semantics of given source program.

The information in the symbol table is entered in the lexical analysis and syntax analysis phase, however, is used in later phases of Compiler (Semantic analysis, intermediate code generation, code optimization and code generation). Intuitively a symbol table maps names into declarations

- Purposes:-

- It is used to store the name of all entities in a structured form at one place
- It is used to implement type checking by verifying assignments and expressions.
- It is used to verify if a variable has been declared.
- It is used to determine the scope of a name.

Error Handler:-

- Report the presence of errors clearly and accurately
- Recover from each error quickly
- Adds minimal overhead to the processing i.e. the correct running programs should not face any processing issue

Q 7>
b>

T→ XR | X

R→ bXR | bX

X→ WbX | W

W→ L* W | L

L→ id

The grammar is not an operator grammar. So we need to convert it to an operator grammar. Now converting it, we get.

T→ XbT | XbX | X

X → WbX | W

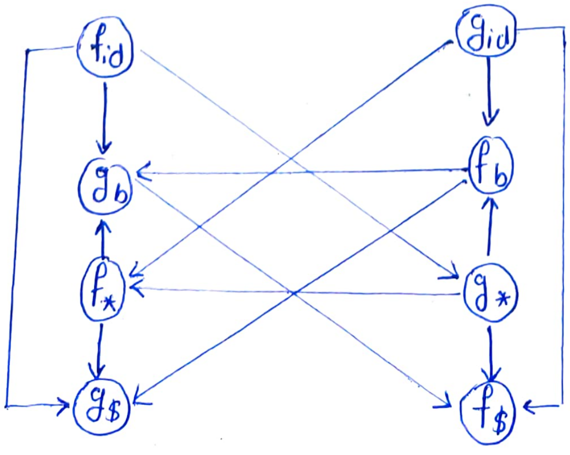
W → L* W | L

L → id

(∵ T→ XR = T→ XbXR = T→ XbT)

Here, the operators are 'b' and '*'. From the grammar, it is evident that * is having higher priority then 'b'. Hence we now construct the operator precedence table.

		id	b	*	$
g	id	–	·>	·>	·>
	b	<·	<·	<·	·>
	*	<·	·>	<·	·>
	$	<·	<·	<·	–
		f			



Solution 2> b> Continue.

The longest path in diagram are

$f_{id} \to g_* \to f_b \to g_b \to f_\$$ and

$g_{id} \to f_* \to g_h \to f_b \to g_\$$

	id	*	b	$
f	5	3	1	0
g	~~5~~4	4	2	0

This is the final function table.

——x——

Solution 3>

Given SDT:

S→ X+Y+Z { S.val = X.val + Y.val + Z.val }

X→ num { X.val = num.val }

Y→ num { Y.val = num.val }

Z→ num { Z.val = num.val }

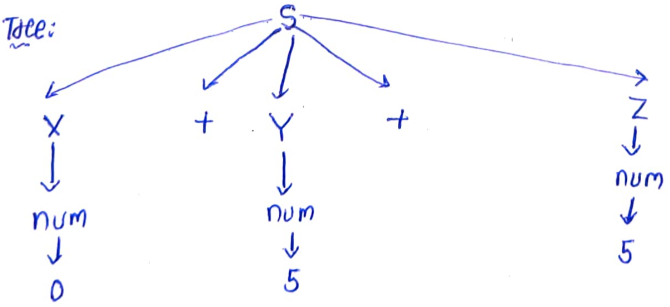
My Roll No: ⇒ 1906055

Last 3 digits ⇒ 055

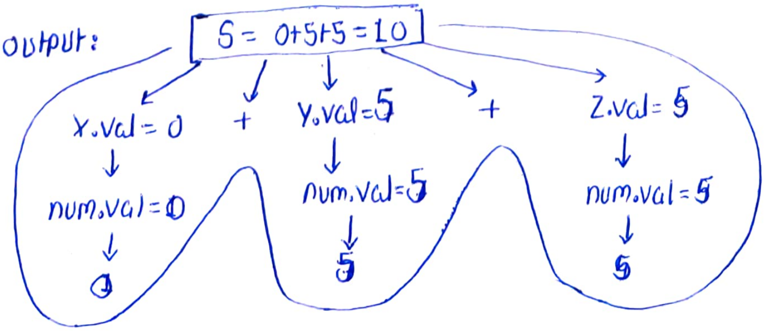
[x=0 , y=5, z=5]

Input String ⇒ 0+5+5

Syntax Tree:



output:



Output = "0+5+5"
= 10

Solution 4)

a)

$Y = W + X$ ---(1)

$Z = Y$ --- (2)

$Y = Y - V$ ---(3)

$W = Z - V$ ---(4)

$X = X * V$ ----(5)

$X = Z / X$ --- (6)

Since, Z=Y and in next step we have Y=Y-V

But in (4) W=Z-V which can be replaced with W=Y as (Z-V and Y) value are same.

⇒ $Y = W + X$
$Z = Y$
$Y = Y - V$
$W = Y$
$X = X * V$
$X = Z / Y$

Since, W= Y and W is not used further in the code. So basically W=Y is Dead Code Elimination. Hence Optimized Code.

$Y = W + X$
$Z = Y$
$Y = Y - V$
$X = X * V$
$X = Z / Y$

Hence, this is optimized code.

Name: Lakhan Kumawat
Roll No: 1906055
Branch: CSE-1
Course: Compiler Design

Course Code: CS5404
Exam Date: 17/12/21
Page No: __

Solution 4>

b> 1> Quadruples.

In quadruples representation each instruction is splitted into the following 4 different fields.

→ op, arg1, arg 2, result.

Advantages.

-> easy to rearrange code for global optimization

→ one can quickly access value of temporary variables using symbol table.

Disadvantages.

→ Contain lot of temporaries.

→ Temporary variable creation increases time and space complexity.

2> Triples. This representation doesn't make use of extra temporary variable to represent a single operation instead when a reference to another triple's value is needed.

Advantages

-> It consists of of only three fileds op, arg1 and arg2.

Disadvantages.

- Temporaries are implicit and difficult to rearrange code.
- It is difficult to optimize because optimization involves moving intermediate code

3> Indirect Triples.

Advantages.

- It is similar in utility as compared to quadruple but requires less space.
- Temporaries are implicit and easier to rearrange.

Disadvantages.

- It make use of pointer to the listing of all references to computation which is made separable and stored.

Name: Lakhan Kumawat
Roll No: 1906055
Branch: CSE-1
Course: Compiler Design

Course Code: CS5404
Exam Date: 17/12/21
Page NO: __

Solution 4) b) The expression is $(w+x) * (y+z) + (w+x+y)$ ⑥

So three address code is

$t_1 = w+x$ --(1)
$t_2 = y+z$ ---(2)
$t_3 = t_1 * t_2$ ---(3)
$t_4 = t_1 + y$ ---(4)
$t_5 = t_3 + t_4$ ---(5)

∴ Quadruple is as follows:

	op	arg1	arg2	result
1	+	w	x	t_1
2	+	y	z	t_2
3	*	t_1	t_2	t_3
4	+	t_1	y	t_4
5	+	t_3	t_4	t_5

Triple is as follows:-

	op	arg1	arg2
1	+	w	x
2	+	y	z
3	+	1	2
4	+	1	Z
5	+	3	4

Indirect triple is as follows.

(100)	1
(101)	2
(102)	3
(103)	4
(104)	5

Index Table

op	arg1	arg2
+	w	x
+	y	z
*	1	2
+	1	y
+	3	4

Name: Lakhan Kumawat

Roll No: 1906055

Branch: CSE-1

Course: Compiler Design

Course Code: CS5404

Exam Date: 17/12/21

Page No: __

Solution 5> a> Control Flow Analysis (CFA) helps us to understand the structure of control flow graphs.

- Determine the loop structure of CFGs.
- Compute dominators
- Compute control dependence
- Compute dominance frontiers - Construction of static single assignment form.

Solution 5>

b> To Derive Three-address Code of the following High-level Code and apply the CFA to identify loop in derived TAC.

```
main() {
        int roll_no[]= {x,y,z}
        int sum=0;
       for(int i=0; i<3;i++)
          {sum += roll_no[i];
          }
        if (sum < 100)
           printf ("CSE-I Student");
        else
           pritf ("CSE-II Student");
          }
```

My Roll No:- 1906055

x=0, y=5, z=5

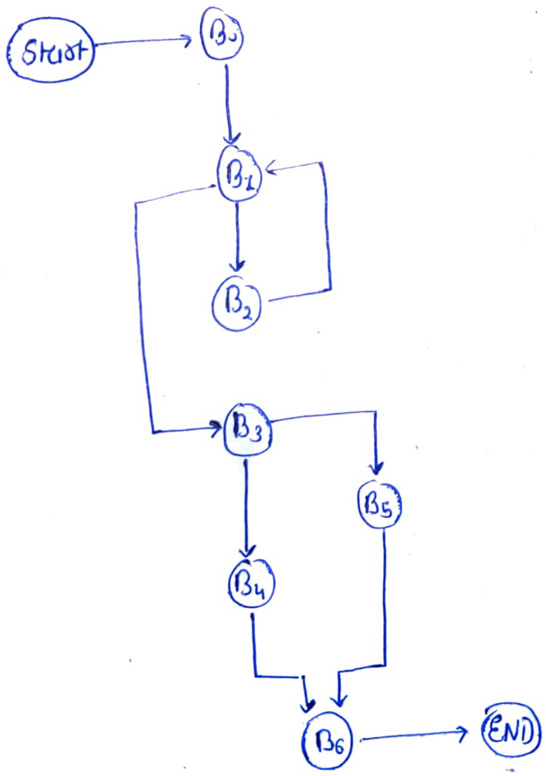
Three Address Code.

```
[ 1.   rollno[0] = 0;      ]
[ 2.   roll no [1] = 5;    ]
[ 3.   roll no [2] = 5;    ]  B0
[ 4.      i=0;             ]
[ 5.     sum = 0;          ]
[ 6.  if (i>= 3) goto 12 . ] B1
```

Name: Lakhan Kumawat
Roll No: 1906055
Branch: CSE-1
Course: Compiler Design

Course Code: CS5404
Exam Date: 17/12/21
Page No: __

```
[  7.  R=R + roll-no[i];          ]
   8.  Sum=R ;                    ]  B2
   9.  t2 = i+1;                  ]
  10.   i=t2;                     ]] B.
[ 11.   goto(6)
[ 12.  If (sum≥100) goto 15 ] B3
[ 13.  print ("CSE-I Student"); ] B4
[ 14.  goto 16
[ 15.  print ("CSE-II Student"); ] B5
[ 16.  goto Calling program; ] B6
```



Start
B_0
B_1
B_2
B_3
B_5
B_4
B_6
END