Name - Arnod Dhopavkar
Class - TE11
Roll No - 33304

DATE __/__/____
PAGE NO _____

SP UT-2

Q.2) c) Generate 3 address code for the statement

while a < b and not c == d do

{

if x < y then

z = w(a) + b ;

else

z = w(a) + z ;

a = a+1 ;

}

Ans) var 1 = a < b

var 2 = c != d

var 3 = x < y

L1 : if (var1 && var 2) go to L3

go to L3

L2 : if (var 3) z = w(a) + b ; a = a+1 ;

z = w(a) + z ; a = a+1 ;

L3 :

Q.3) a) Discuss code generation issues.

Ans) 1. Input to code generator - The intermediate code generation phase produces output in the form of intermediate representation (IR) in the front end of the compilation process.

2. Target program - The output of code generator is output machine code. Target code may be absolute machine language, relocatable machine language, assembly language.

3. Address mapping - It is used for mapping between IR to the address in the machine code.

4. Memory management - The front end and code generator does the mapping btw the symbol / variable names in the program.

Q.3)a) Explain flow graph.

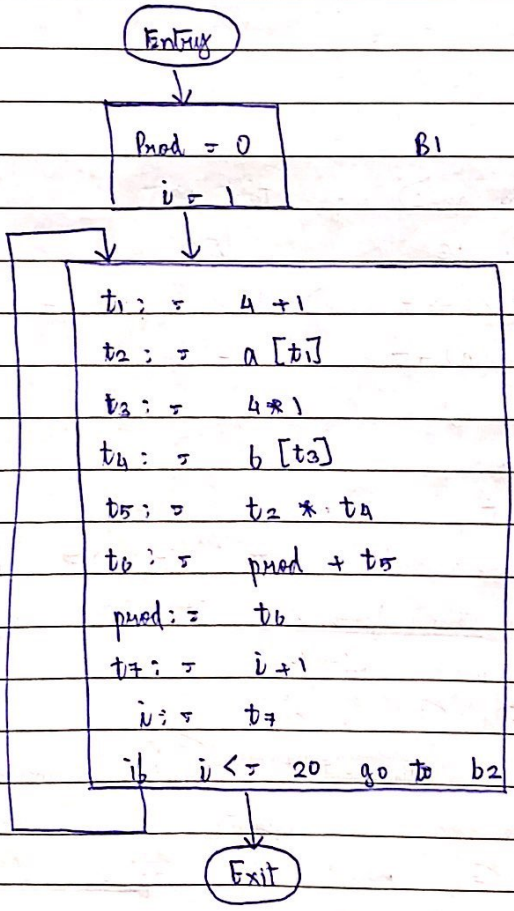
Ans) 1. Basic blocks generated by code generator in a program can be represented by means of control flow graphs

2. In control flow graphs, basic blocks are represented as nodes and the edges indicate the sequence of blocks

3. It represents the passing of program control between various blocks.

4. It is an important tool that helps in the process of optimization by locating any unnecessary loops in the program.

Eg - Flow graph for vector addition



Q.3)b) Explain the following machine independent optimization techniques.

Ans) Machine independent optimization techniques are used to improve the intermediate code, to get a better target code. The part of the code which is transformed here does

not involve any absolute memory location or on CPU registers

The process of intermediate code generation introduces much inefficiency like - using ~~value~~ variables instead of constants → extra copies of variables → repeated evaluation of data.

1. Dead code elimination :
One or more code statements which are either never executed or are unreachable ; or if executed → their output is never used.

2. Strength reduction :
It is used to replace the high strength operator by low strength. An induction variable is used in loop for this kind of statement.

Before reduction →

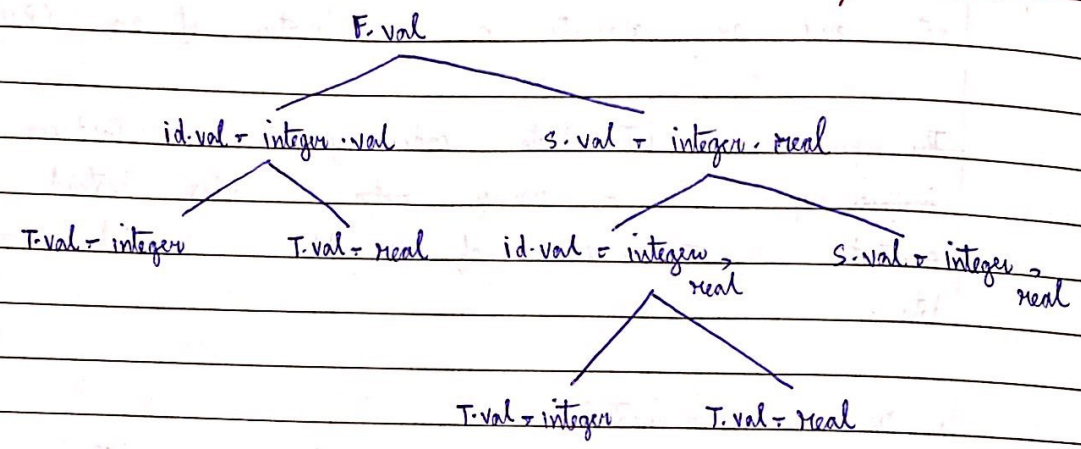
```
i = 1 ;
while (i < 10)
        y = i * 4 ;
```

After reduction →

```
i = 1
t = 4
while (t < 40)
{
        y = t
        t = t + 4
}
```

Q.2) a) For the input string A, B, C : REAL construct an annoted parse tree according to the following syntax directed definition

Ans)



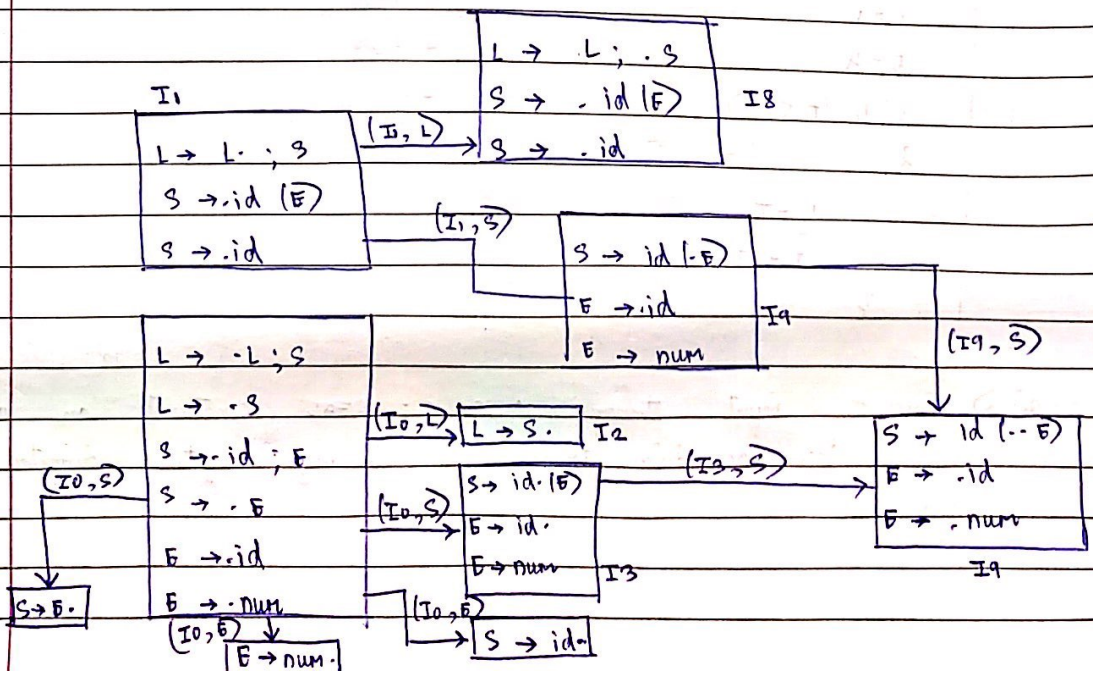
Annotated parse tree

Q.2)b) For the attributes of the syntax directed definition of Q.2)a) indicate whether each attribute is inherited or synthesized using the following table

Ans)

	Inherited	Synthesized
id. entry	~~Yes~~	No
S. entry	Yes	No
T. type	No	Yes

Q.1)a) Consider the following grammar. Construct DFA of LR(0) items and SLR(1) parsing table.



	Inherited	Synthesized
id. entry	Yes	No
S. entry	Yes	No
T. type	No	Yes

Q.1) b) Consider the following grammar

exp → id | (exp - list + op)

op → + | - | *

exp - list → exp - list exp | exp

Remove the left recursion and create first and follow sets for non-terminals of the resulting grammar

Ans) Grammar after removing left recursion →

exp → id

op → + | - | *

exp - list → exp exp - list'

exp - list' → exp exp - list' | ∈

	Follow	First
exp	$	(, id, ∈
exp - list	(, id, ∈	∈, id, (
exp - list'	id, (, ∈	id, ∈, (
op	id, ∈	+, -, *