SPCC

* Postfix Notation

1) a+b*c ⇒ a+bc*
⇒ abc*+

2) a*b+c ⇒ ab*+c ⇒ ab*c+

3) a+b*c-d ⇒ a+bc*-d ⇒ abc*+-d ⇒ abc*+d-

4) a*b+c/d ⇒ ab*+cd/ ⇒ ab*cd/+

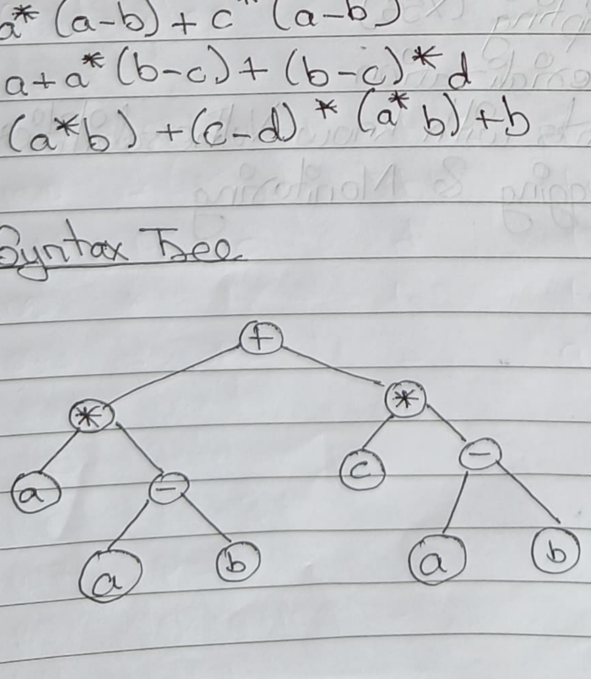
5) a*(b+c/d) ⇒ a*(b+cd/) ⇒ a*(bcd/+)
⇒ a(bcd/+)* ⇒ abcd/+*

6) a*(b+c)/d ⇒ a*(bc+)/d ⇒ ~~a*(bc+d~~/ a(bc+)*/d
⇒ abc+*d/

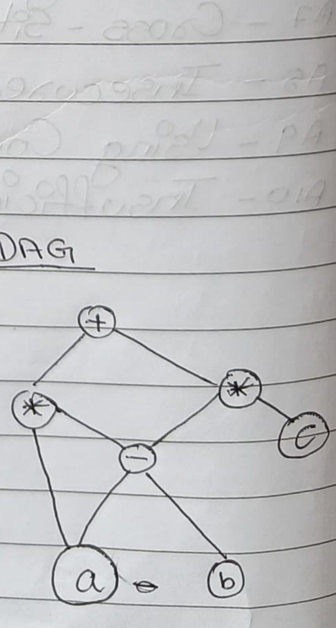
* Syntax tree and DAG

1) a*(a-b)+c*(a-b)
2) a+a*(b-c)+(b-c)*d
3) (a*b)+(c-d)*(a*b)+b

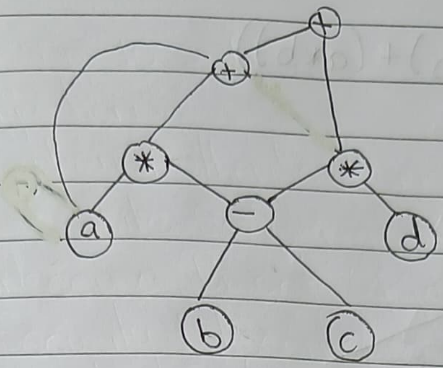
Ans 1) Syntax Tree



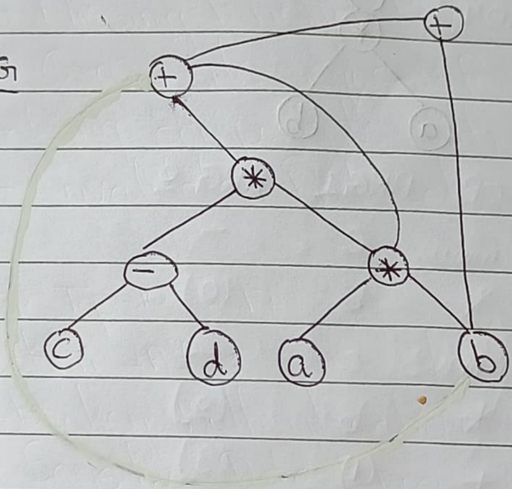
DAG



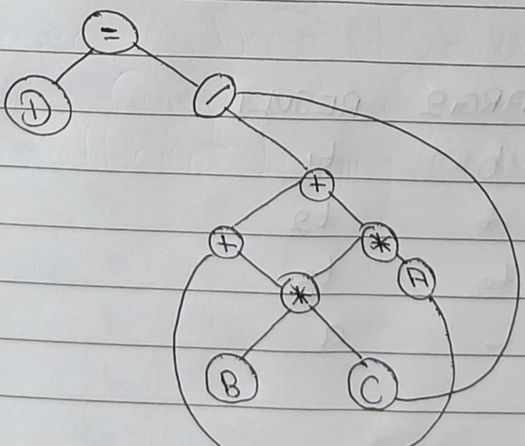
Ans 2) DAG



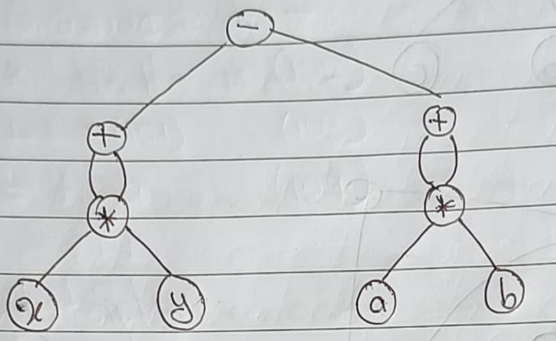
Ans 3) DAG



4) D=((A+B*C)+(A*B*C))/C



5) $((x*y)+(x*y)) - ((a*b)+(a*b))$



* 3 Address Code Form

1) $a = (e-b) * (c+d)$

→ $t_1 = e - b$

$t_2 = c + d$

$t_3 = t_1 * t_2$

~~$a = t_3$~~ $a = t_3$

* Quadruples

	OP	ARG1	ARG2	RESULT
100	-	e	b	t_1
101	+	c	d	t_2
102	*	t_1	t_2	t_3
103	=	t_3	-	a

* Triples

	OP	ARG1	ARG2
100	-	e	b
101	+	c	d
102	*	(100)	(101)
103	=	a	(102)

* Indirect Triple

0	100
1	101
2	102
3	103

	OP	ARG1	ARG2
100	-	e	b
101	+	c	d
102	*	(0)	(1)
103	=	a	(2)

Q) $a = b * -c + b * -c$ [$-c$ = Unary Minus]

→ $t_1 =$ ~~$-c$~~ uminus c

$t_2 = b * t_1$

$t_3 =$ ~~$t_2 + t_2$~~ uminus c

~~$a = t_3$~~ $t_4 = b * t_3$

$t_5 = t_2 + t_4$, $a = t_5$

Quadruples

	OP	ARG1	ARG2	RESULT
100	uminus	c	-	t_1
101	*	b	t_1	t_2
102	uminus	c	-	t_3
103	*	b	t_3	t_4
104	+	t_2	t_4	t_5
105	=	t_5	-	a

* Triples

	OP	ARG1	ARG2
100	-	e	b
101	+	c	d
102	*	(100)	(101)
103	=	a	(102)

* Indirect Triple

0	100
1	101
2	102
3	103

	OP	ARG1	ARG2
100	-	e	b
101	+	c	d
102	*	(0)	(1)
103	=	a	(2)

Q) $a = b * -c + b * -c$ [$-c$ = Unary Minus]

→ $t_1 =$ ~~-c~~ uminus c

$t_2 = b * t_1$

$t_3 =$ ~~$t_2 + t_2$~~ uminus c

~~$a = t_3$~~ $t_4 = b * t_3$

$t_5 = t_2 + t_4$, $a = t_5$

Quadruples

	OP	ARG1	ARG2	RESULT
100	uminus	c	-	t_1
101	*	b	t_1	t_2
102	uminus	c	-	t_3
103	*	b	t_3	t_4
104	+	t_2	t_4	t_5
105	=	t_5	-	a

* Triples

	OP	ARG1	ARG2
100	-	e	b
101	+	c	d
102	*	(100)	(101)
103	=	a	(102)

* Indirect Triple

0	100
1	101
2	102
3	103

	OP	ARG1	ARG2
100	-	e	b
101	+	c	d
102	*	(0)	(1)
103	=	a	(2)

Q) $a = b* -c + b* -c$ [$-c$ = Unary Minus]

→ t_1 = ~~-c~~ uminus c

$t_2 = b*t_1$

t_3 = ~~t_2+t_2~~ uminus c

~~$a = t_3$~~ $t_4 = b*t_3$

$t_5 = t_2+t_4$, $a = t_5$

Quadruples

	OP	ARG1	ARG2	RESULT
100	uminus	c	-	t_1
101	*	b	t_1	t_2
102	uminus	c	-	t_3
103	*	b	t_3	t_4
104	+	t_2	t_4	t_5
105	=	t_5	-	a

→ triples:-

	OP	ARG 1	ARG 2
100	uminus	c	–
101	*	b	(100)
102	uminus	c	–
103	*	b	(102)
104	+	(101)	(103)
105	=	a	(104)

→ indirect triples

			OP	ARG 1	ARG 2
0	100	100	uminus	c	–
1	101	101	*	b	(0)
2	102	102	uminus	c	–
3	103	103	*	b	(2)
4	104	104	+	(1)	(3)
5	105	105	=	a	(4)

Q.1) if (x<y)
    a=b+c*3
else
    p=q+r

→ 100 if (x<y) goto 102
101 else goto 106
102 $t_1 = c*3$
103 $t_2 = b+t_1$
104 $a = t_2$
105 goto 109
106 $t_3 = q+r$
107 $p = t_3$
108 goto 109
109 STOP

2)
```
while (i<=n) {
    sum = sum + i;
    i++;
}
→ 100 if (i<=n) go to 102
  101 else goto 107
  102 t1 = sum+i
  103 sum = t1
  104 t2 = i+1
  105 i = t2
  106 go to 100
  107 STOP
```

3)
```
do {
    sum = sum+i;
    i++;
} while (i<=n);
→ 100 t1 = sum+i
  101 sum = t1
  102 t2 = i+1
  103 i = t2
  104 if (i<=n) go to 100
  105 else go to 106
  106 STOP
```

4)
```
while (a<b) do
    if (c<d)
        x = y+z
→
  100 if (a<b) go to 102
  101 else go to ~~106~~ 107
  102 if (c<d) go to 104
  103 else go to ~~106~~ 107
  104 t1 = y+z
  105 x = t1
  106 ~~STOP~~ go to 100
  107 STOP
```

***

5)
```
while (a<b) do
    if (a>b)
        x = a+b
    else
        x = a-b
→ 100 if (a<b) goto 102
  101 else go to 110
  102 if (a>b) go to 104
  103 else go to 107
  104 t1 = a+b
  105 x = t1
  106 go to 100
  107 t2 = a-b
  108 x = t2
  109 go to 100
  110 STOP
```

6)
```
for (i=1; i<=n; i++){
        f=f*i;
}
```

→
```
100  i=1
101  if (i<=n) go to 103
102  else go to 108
103  t1=f*i
104  f=t1
105  t2=i+1
106  i=t2
107  go to 101
108  STOP
```

7) If A<B and C<D then t=1 else t=0

8) (a*b)+(c+d)-(a+b+c+d)

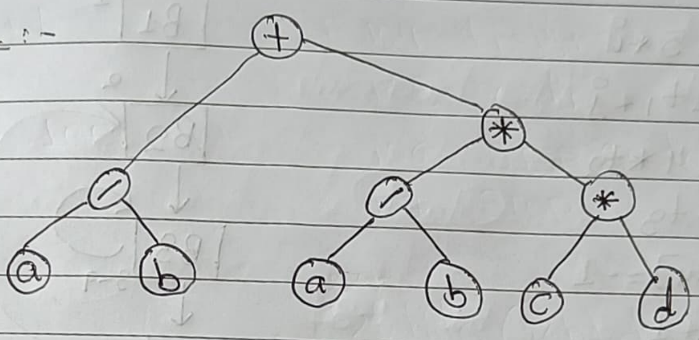
Ans 7)
```
100  if (a<b) go to 102
101  go to 107
102  if (c<d) go to 104
103  go to 106
104  t=1
105  go to 107
106  t=0
107  STOP
```

Ans 8)
```
100  t1=c+d
101  t2=b+t1
102  t3=a+t2
103  t4=c+d
104  t5=a*b
105  t6=t5+t4
106  t7=t6-t3
```

Q) Write postfix notation. Draw syntax tree and DAG and also compute 3AC for the expression
(a/b)+(a/b)*(c*d)

→ Postfix notation :- (a/b)+(a/b)*(cd*)
(ab/)+(ab/)*(cd*)
(ab/)+(ab/cd**)
(ab/ab/cd**+)

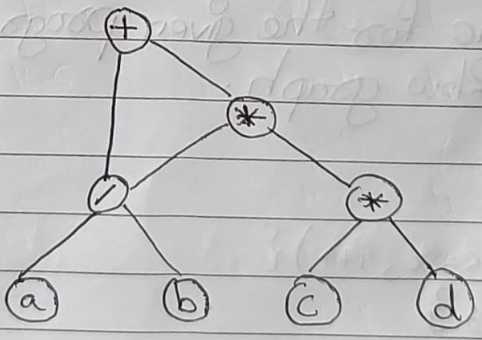
→ Syntax Tree :-



→ 3AC :-

100 $t_1 = a/b$
101 $t_2 = a/b$
102 $t_3 = c*d$
103 $t_4 = t_2 * t_3$
104 $t_5 = t_1 + t_4$

→ DAG :-



classmate

Date

Page

* Basic Block and Flow Graph

Q) Consider the intermediate code given below. Draw basic block and flow graph and find the number of nodes and edges in control-flow-graph constructed for the below code.

→ 1) $i = 1$

→ 2) $j = 1$

→ 3) $t_1 = 5 * i$

4) $t_2 = t_1 + j$

5) $t_3 = 4 * t_2$

6) $t_4 = t_3$

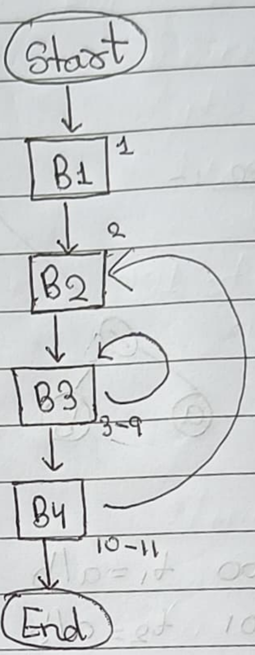
7) $a[t_4] = -1$

8) $j = j + 1$

9) if ($j <= 5$) go to (3)

→ 10) $i = i + 1$

11) if ($i < 5$) go to 2



No. of nodes = 4 [Except Start & End]

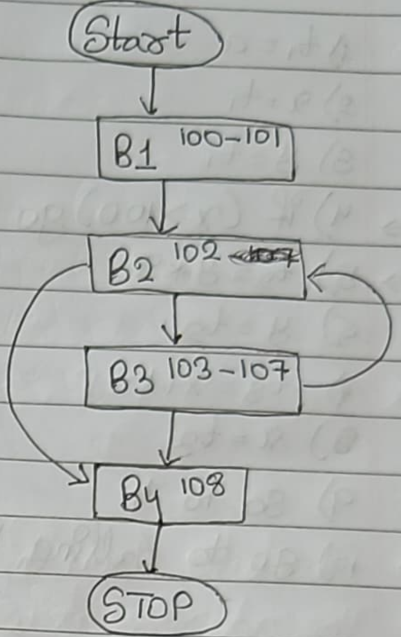
No. of edges = 7

Q) Compute 3AC for the given program and draw basic block and flow graph.

```
fact (x) {
  int f = 1
  for (i=2; i<=x; i++) {
      f = f * i; }
    return f;
}
```

Jab initialise होगा तो Direct लिखना है और जब Assigment होगा तो Temporary मे store करके लिखना है।

```
→ 100  f = 1
  101  i = 2
→ 102  if i>x go to 108
→ 103  t1 = f * i
  104  f = t1
  105  t1 = i+1
  106  i = t1
  107  go to 102
→ 108  Go to Calling Progoram
```

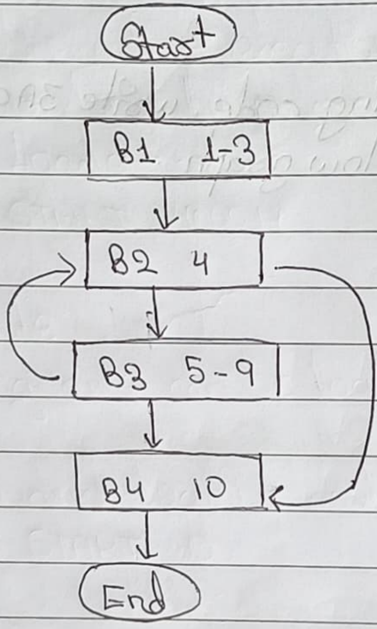


Q) Consider the following code, write 3AC and draw basic block and flow graph.

```
int func(int x) {
  int x, y;
  x = a;
  y = a;
  while (a<100) {
     y = y*x;
     x = x+1;
  }
  return y;
}
```

```
→ 1) t1 = a                    9) t4 = x+1
  2) x = t1                    10) x = t4
  3) t2 = a                 → 11) go to 5
  4) y = t2                 → 12) Go to Calling Program
→ 5) if (x<100) go to 7
→ 6) go to 12
→ 7) t3 = y*x
  8) y = t3
```

```
→   1) t1 = a
    2) x = t1
    3) y = t1
→   4) if (x>100) go to 10
→   5) t2 = y * x
    6) y = t2
    7) t3 = x+1
    8) x = t3
    9) go to 4
→   10) go to calling Program
```



Q)
```
prod=0; i=1;
do {
prod = prod + a[i]*b[i]
  i = i+1;
} while (i<=10);
```

→ 1) prod = 0

→ 2) i = 0

→ 3) $t_1 = 4 * i$

4) $t_2 = a[t_1]$

5) $t_3 = 4 * i$

6) $t_4 = b[t_3]$

7) $t_5 = t_2 * t_4$

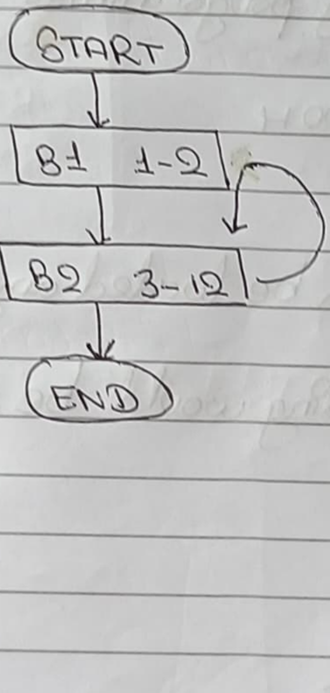
8) ~~prod = pro~~ $t_6 = prod + t_5$

9) prod = t_6

10) $t_7 = i + 1$

11) $i = t_7$

12) if ($i <= 10$) go to 3


START
B1 1-2
B2 3-12
END