6(A) i)

$G_T = (\{S, A, B\}, \{a, b\}, P, S)$

where P consists of

$S \rightarrow aB / bA$

$A \rightarrow a / aS / bAA$

$B \rightarrow b / bS / aBB$

"bbaaba"

Leftmost derivation

$S \rightarrow bA$	$(S \rightarrow bA)$
$\rightarrow bbAA$	$(A \rightarrow bAA)$
$\rightarrow bbaA$	$(A \rightarrow a)$
$\rightarrow bbaaS$	$(A \rightarrow aS)$
$\rightarrow bbaabA$	$(S \rightarrow bA)$
$\rightarrow bbaaaba$	$(A \rightarrow a)$

Rightmost derivation

$S \rightarrow bA$	$(S \rightarrow bA)$
$\rightarrow bbAA$	$(A \rightarrow bAA)$
$\rightarrow bbAa$	$(A \rightarrow a)$
$\rightarrow bbaSa$	$(A \rightarrow aS)$
$\rightarrow bbaaBa$	$(S \rightarrow aB)$
$\rightarrow bbaaba$	$(B \rightarrow b)$

6 A) ii)

Operator Precedence Parser

- It is a bottom-up parser that interprets an operator-precedence program.
- Example: Most calculators use operator precedence parsers to convert from the human readable infix notation.

The operator precedence parsing technique can be applied to operator grammars

Operator grammar are defined as grammars with following properties

1. No epsilon in the right hand side of any production
2. No adjacent non-terminal in the right hand side of any production.

This property enables the implementation of efficient operator-precedence parsers These parsers rely on 3 precedence relations.

Relation	Meaning
a <· b	a yields precedence to b
a =· b	a has the same precedence as b
a ·> b	a takes precedence over b

SIGNATURE: Amey

For example,

the following operator precedence relation can be introduced for simple expression.

	id	+	*	$
id		$\dot{>}$	$\dot{>}$	$\dot{>}$
+	$\lessdot$	$\dot{>}$	$\lessdot$	$\dot{>}$
*	$\lessdot$	$\dot{>}$	$\dot{>}$	$\dot{>}$
$	$\lessdot$	$\lessdot$	$\lessdot$	

Difference between operator precedence parsing and recursive decent parsing

6 (B) ii)

Sol?:

Three address code for a given expression

```
while (a<b) do
if (c<d) then
x = y + 2
else
x = y - 2
```

```
L1 : while (a<b) goto L2
     goto last

L2 : if (c<d) goto L3
     Goto L4:

L3 : t1 = y
     t2 = 2
     t3 = t1 + t2
     x = t3

L4 : t1 = y
     t2 = 2
     t3 = t1 - t2
     x = t3
     last
```