WRITING EXPANDTEMPLATE SCRIPT

1.1 FOR LOOPS

1.1.1 Syntax

```
#FOR VARIABLE=INITIAL:STEP:FINALVALUE
   STATEMENTS
#END_FOR
```

- ***VARIABLE*** is name of variable you want to increment or decrement
- ***INITIAL, STEP, FINALVALUE*** can be a constant or a formula

Formula can contain following operations/symbols **- (,)**, +,-,*,/ ,variable names ,constants

1.1.2 Working

Statements between **#FOR** and **#END_FOR** will repeat (**FINALVALUE-INITIAL**)/**STEP** times and in each repeated portion Variable name will be replaced by the calculated value ie, in first port Variable name will be **INITIAL**, in second portion Variable name will be **INITIAL**+**STEP** and so on.

Nested FOR loops can be written

1.1.3 Example

INPUT FILE

```
#FOR Srno=1:1:3
<<Srno>>
#END_SEQ
```

OUTPUT FILE

```
1
2
3
```

1.2 WRITING IF STATEMENTS

1.2.1 Syntax

```
IF VARIABLE< VALUE
   STATEMENTS
#END_IF
```

OR

```
IF VARIABLE> VALUE
   STATEMENTS
#END_IF
```

OR

```
IF VARIABLE<= VALUE
   STATEMENTS
#END_IF
```

OR

```
IF VARIABLE>= VALUE
   STATEMENTS
#END_IF
```

OR

```
IF VARIABLE== VALUE
   STATEMENTS
#END_IF
```

OR

IF VARIABLE!= VALUE
   STATEMENTS
#END_IF

- ***VALUE*** can be a constant or a formula
- ***VARIABLE*** is name of variable you want to check for

1.2.2 Working

Statement between **#IF** and **#END_IF** will not come in the output if *IF condition* is not satisified

Nested IF can be written

1.3 WRITING SEQUENCES

1.3.1 Syntax

#SEQ1 VARIABLE=INITIAL:STEP:FINALVALUE
   STATEMENTS
#END_SEQ1

OR

#SEQ1 VARIABLE=INITIAL:STEP/AFTERLINES:FINALVALUE
   STATEMENTS
#END_SEQ1

- ***STEP/AFTERLINES*** means ***"STEP"*** increment/decrement after every ***" AFTERLINES"*** *number of* lines
- ***VARIABLE*** is name of variable you want to increment or decrement
- ***INITIAL, STEP, FINALVALUE, AFTERLINES*** can be a constant or a formula
- In ***STEP, & AFTERLINES*** division operation cannot be used

Formula can contain following operations/symbols - (,), +,-,*,/ ,variable names ,constants

- If you are writing overlapping sequences give different names for sequences as given below

example

#SEQ1 VARIABLE=1:2/1:10
   STATEMENTS

```
#SEQ2 VARIABLE=10:1:1
   STATEMENTS
#END_SEQ2
#END_SEQ1
```

#RESTART_SEQ* - will restart sequence from initial value
#INCREMENT_SEQ* -will increment value.(In any case value will not be incremented more than **STEP**)

1.3.2 Working

In statement between **#SEQ** and **#END_SEQ VARIABLE** will replaced by value from **INITIAL** to **FINALVALUE** incremented/decremented by **STEP** after every ***" AFTERLINES"*** *number of* lines.

Only the line containing **VARIABLE** will be accorded for incrementing/decrementing the value.

After reaching **FINALVALUE** sequence restarts.

1.3.3 Example

Example 1

INPUT FILE

```
#SEQ1 Srno=1:1:3
<<Srno>>
<<Srno>>
<<Srno>>
<<Srno>>
#END_SEQ1
```

OUTPUT FILE

```
1
```

```
2
3
1
```

Example 2

INPUT FILE

```
#SEQ1 Srno=1:1:3
<<Srno>>
<<Srno>>
#RESTART_SEQ1
<<Srno>>
<<Srno>>
#END_SEQ1
```

OUTPUT FILE

```
1
2
1
2
```

Example 3

INPUT FILE

```
#SEQ1 Srno=1:1/2:3
<<Srno>>
<<Srno>>
<<Srno>>
#RESTART_SEQ1
<<Srno>>
```

```
<<Srno>>
<<Srno>>
<<Srno>>
<<Srno>>
#END_SEQ1
```

OUTPUT FILE

```
1
1
2
3
3
4
4
5
```

1.4 WRITING STATEMENTS

1.4.1 Syntax

```
A=<<VARIABLE1>> B=<<VARIABLE2:3>>     result(A+B)=<<VARIABLE1 +VARIABLE2>>   VARIABLE1
```

Here **VARIABLE1** and **VARIABLE2** will be replace by the respective value passed by arguments/value obtained for FOR loop/value obtained from SEQ . Only portion inside << >> will be processed by application rest of the portion is copies as such.

<<VARIABLE2:3>> will be replace by value of **VARIABLE2** with zeros before it to make it a three digit number **(eg; 001)**

if **VARIABLE1** = 2 & **VARIABLE2**=5 then final output will be

```
A=2 B=005    result(A+B)=7    VARIABLE1
```

1.5 WRITING COMMENTS

//* COMMENT1

//* COMMENT2

argument **--include_comments** will print comments in the output file along with processed text.