Pass-I Microprocessor

Name - Kajal Sunil Pagare.
Rollno - 26 Div - B
Class - TE.

Aim - To design Data structure for Microprocessor.

Problem Statement - Design suitable data Structures and implement Pass-I of a two-Pass macro-processor using OOP features in Java.

Theory :

1. Micro processor :

A micro processors, i's a program i's a program that reads a files, and scan them for certains keywords. When a keyword is found, it is replaced by some text.

2. Basic tasks performed by macroProcessor :

A) Recognize macro definations.
B) Save the definations.
C) Recognize call.
D) Expanded calls and substitude argument.

3) macrodefination part.
4) Macro calls expansion.

5. Implementation Logic.
   I. Definition Processing.
   II. Macro Expansion
6. Data structure required for macro definition processing,
   I. Macro Name Table (MNT)
   II. Parameter Name Table (PNTAB)
   III. Keyword parameter Default Table.
   IV. macro Definition Table (MDT)
7. Algorithms / Pseudo code.

Algorithms -
A one-pass macroprocessor that alternate between macro definition and macro expansion algorithms.

Algorithms
```
begin
    EXPANDING := FALSE
    While OPCODE ≠ 'END' do
        begin
            GETLINE
            PROCESSLINE
        end {while}
    end {macro processor}
Procedure PROCESSLINE
begin.
```

MEERA
PAGE NO.:
DATE: / /

search NAMTAB for OPCODE.
if found then
EXPAND
elseif OPCODE = "MACRO" then
DEFINE.
else write source line to expanded
file end {PROCESSIVE}

Algorithm:
procedure EXPAND
begin
EXPANDING := TRUE.
get first line of macro definition
{prototype} from DEFTAB
set up arguments from macro invoca-
tions in ARGTAB
write macro invocation to expand file as
comments.
while not end of macro definition do.
begin
GETLINE.
PROCESSINE.
end {while}
EXPANDING = FALSE
end {EXPAND}
procedure GETLINE.
begin
if EXPANDING then

begin get next line of macro definitions from DEFTAB.
Substitute arguments from ARGTAB for positional notation
end {if}
else
read next line from input file.
end {GETLINE}

Example.

Source.	Expanded source.
STRG MACRO	⋮
STA DATA1	STA DATA1
STB DATA2	STB DATA2
STX DATA3	STX DATA3
MEND	STA DATA1
STRG	STB DATA2
STRG	STX DATA3

Source.	Expended source...
STRG MACRO &a1, &a2, &a3	STA DATA1
STA &a1	STB DATA2
STB &a2	STX DATA3
STX &a3	
MEND	STA DATA4
STRG DA1, DA2, DA3	STB DATA5
STRG DA4, DA5, DA6	STX DATA6

Input

```
MACRO INCR &X &Y &REG1
    ADD REG &Y
     MOVEM &REGI &X.
MEND
        START 100
        READ N1
         READ N2
        INCR N1 N2.
         STOP
          N1 DS1
          N2 DS2
           END
```

```
C:\ABC> javac macro.java
C:\ABC> java macro.
```

```
MACRO INCR &X &Y &REG1
   MOVER   &REG1 &X
   ADD     &REG1 &Y
   MOVEM   &REG1 &X
MEND
   START   100
   READ    N1
   READ    N2
   INCR    N1    N2
   STOP
   N1      DS    1
   N2      PS    2
    END
```

MEERA
PAGE NO.:
DATE: / /

MNT:

INDE	MACRONAME	MDT INDEX
1	INCR	1

ALA:

INDEX	ARGUMENT
#1	&X
#2	&Y
#3	&REG1

MDT:

MACRO	INCR	&X	&Y	&REG1
	MOVER	#3	#1	
	ADD	#3	#2	
MEND	MOVEM	#3	#1	

Conclusion → Thus pass-1 microprocessor is implemented and, MNT, MDT & ALA file is generated.