Practical no: 3

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

Name: Aditi Dinesh Mulay

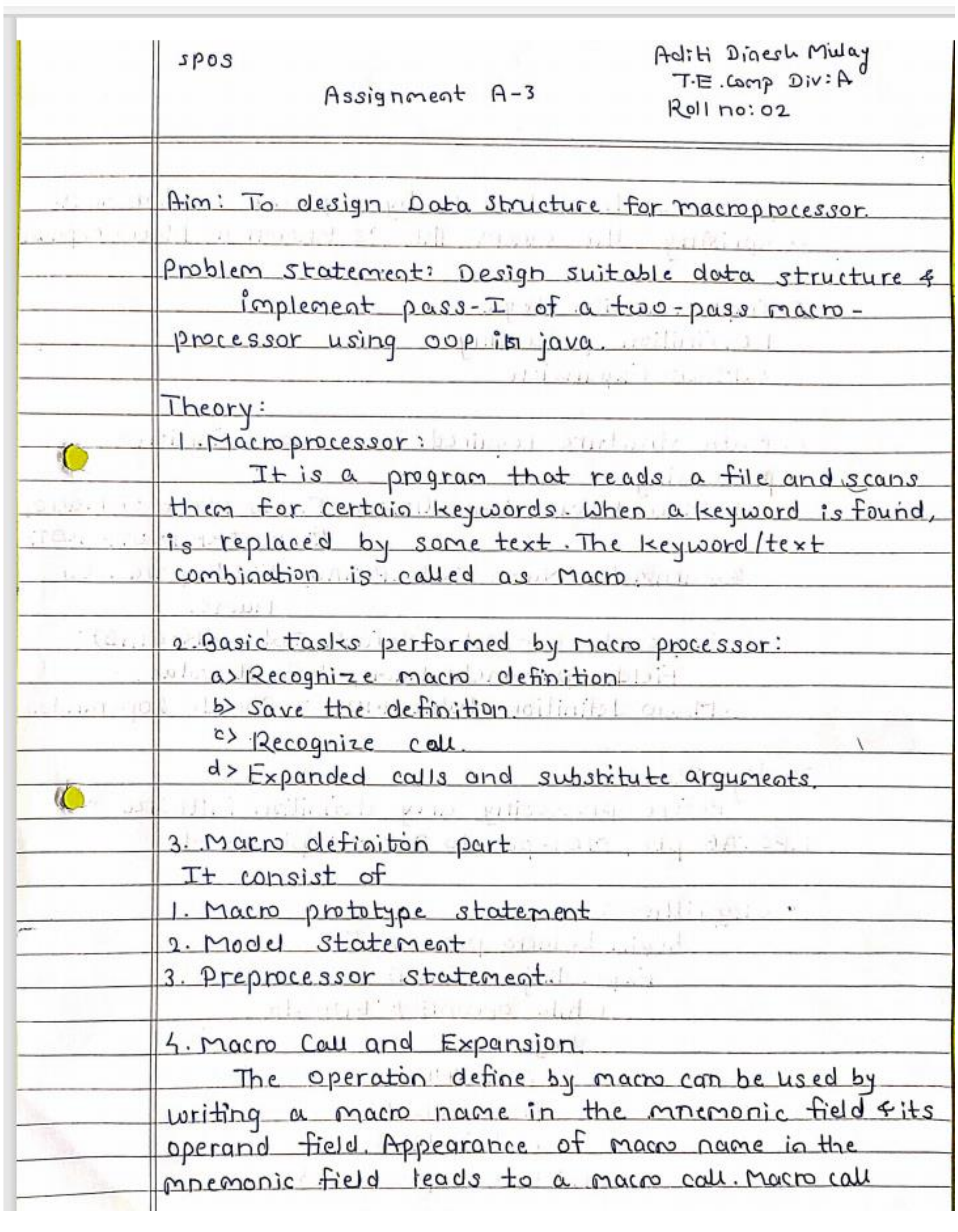
Class: T.E. Computer

Subject: SPOS

Div: A

Roll no: 02

PRN No. 71918146B



SPOS

Aditi Dinesh Mulay
T.E.Comp Div:A
Roll no:02

Assignment A-3

Aim: To design Data Structure for macroprocessor.

Problem statement: Design suitable data structure & implement pass-I of a two-pass macro-processor using OOP in java.

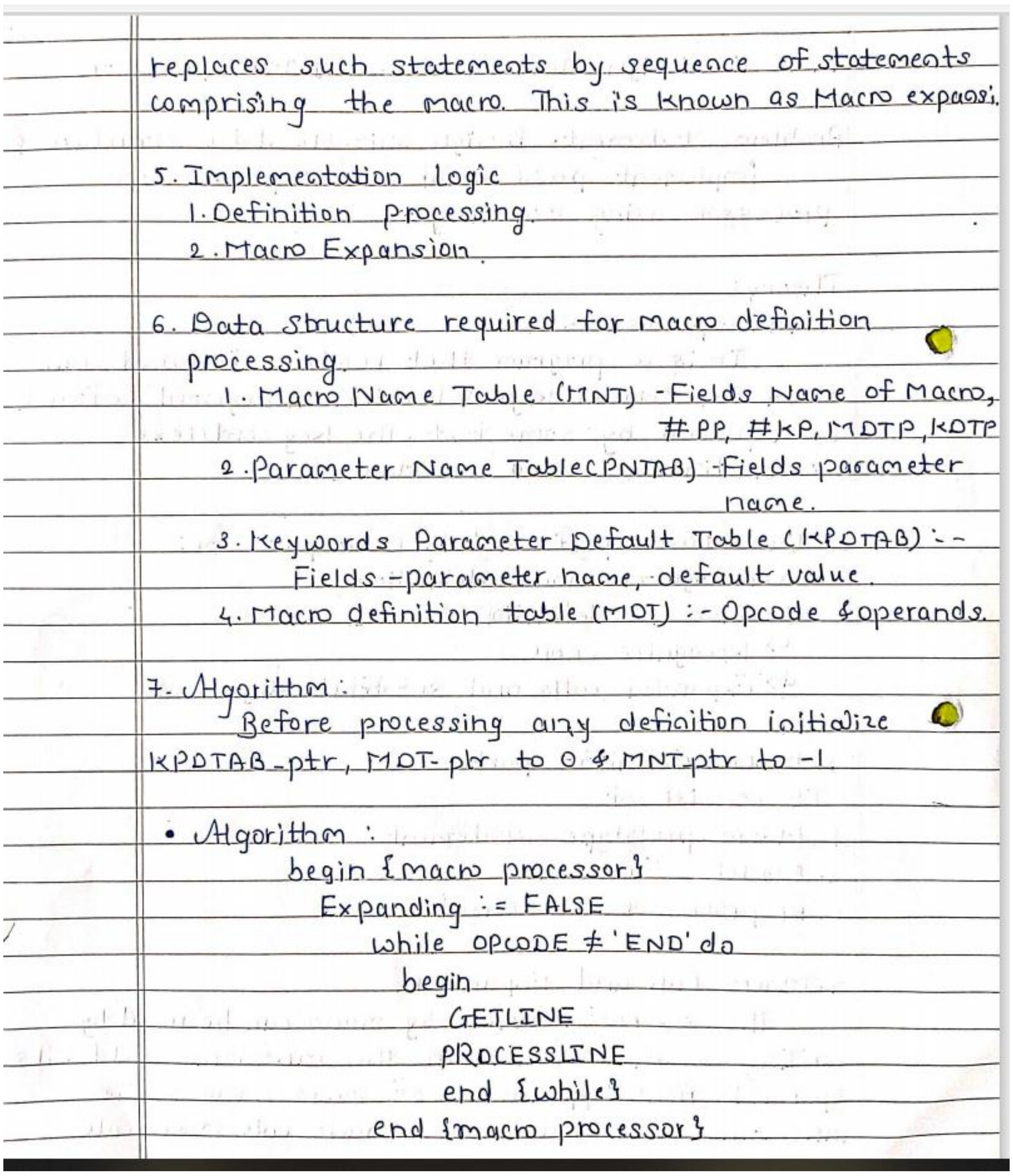
Theory:

1. Macroprocessor:
It is a program that reads a file and scans them for certain keywords. When a keyword is found, is replaced by some text. The keyword/text combination is called as Macro.

2. Basic tasks performed by Macro processor:
a> Recognize macro definition
b> Save the definition.
c> Recognize call.
d> Expanded calls and substitute arguments.

3. Macro definiton part
It consist of
1. Macro prototype statement
2. Model statement
3. Preprocessor statement.

4. Macro Call and Expansion.
The operaton define by macro can be used by writing a macro name in the mnemonic field & its operand field. Appearance of macro name in the mnemonic field leads to a macro call. Macro call



replaces such statements by sequence of statements comprising the macro. This is known as Macro expansi

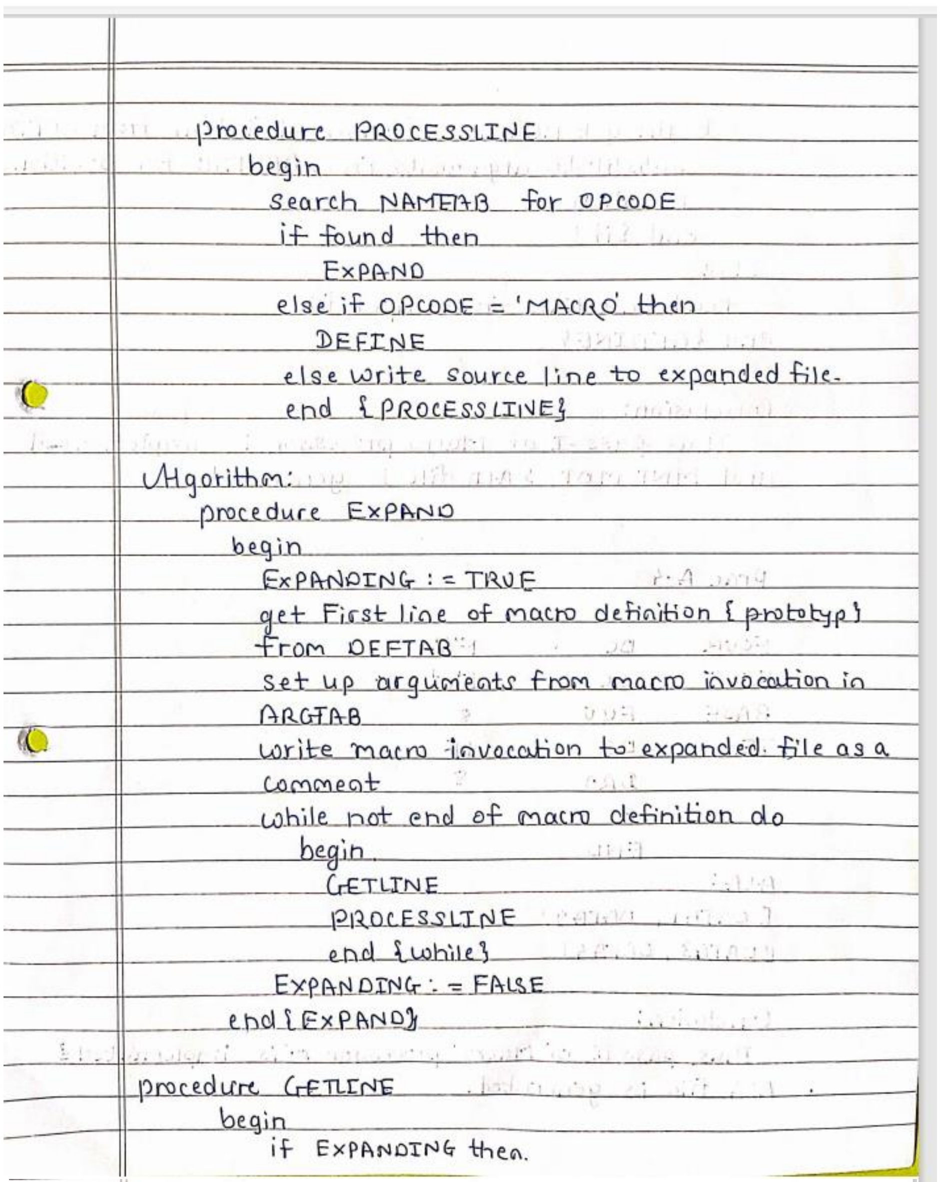
5. Implementation logic
   1. Definition processing.
   2. Macro Expansion.

6. Data Structure required for macro definition processing.
   1. Macro Name Table (MNT) - Fields Name of Macro, #PP, #KP, MDTP, KDTP
   2. Parameter Name Table(PNTAB) - Fields parameter name.
   3. Keywords Parameter Default Table (KPDTAB) :- Fields - parameter name, default value.
   4. Macro definition table (MDT) :- Opcode & operands.

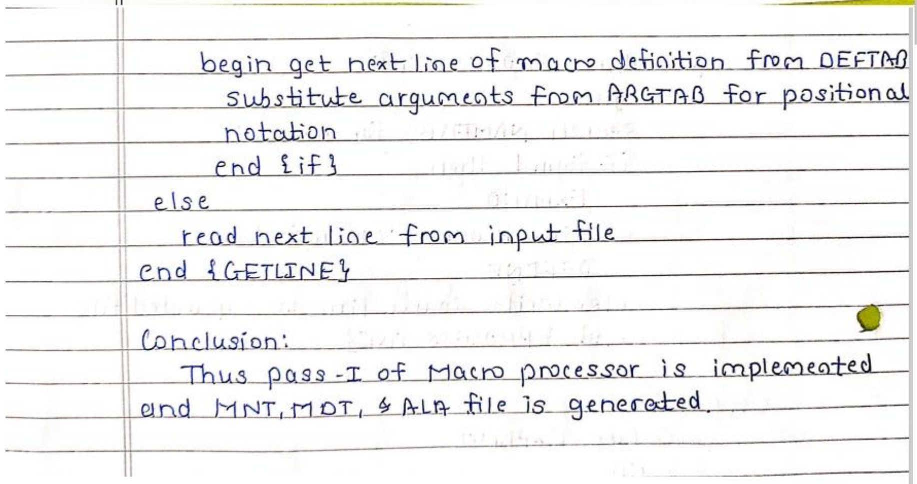
7. Algorithm :
Before processing any definition initialize KPDTAB_ptr, MDT-ptr to 0 & MNT-ptr to -1.

• Algorithm :

```
begin {macro processor}
    Expanding := FALSE
        while OPCODE ≠ 'END' do
            begin
                GETLINE
                PROCESSLINE
            end {while}
        end {macro processor}
```



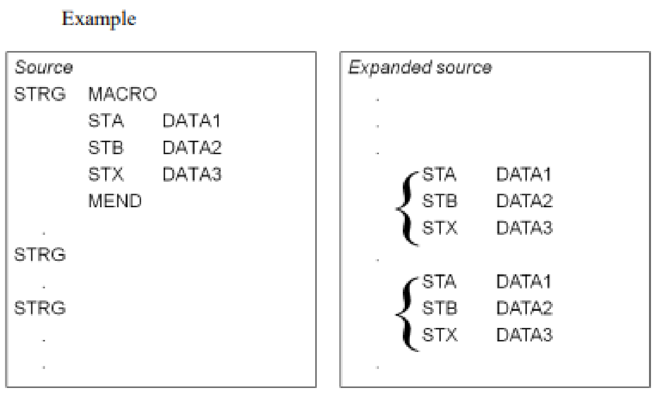
```
procedure PROCESSLINE
   begin
     search NAMTAB for OPCODE
     if found then
        EXPAND
     else if OPCODE = 'MACRO' then
        DEFINE
     else write source line to expanded file.
     end {PROCESSLINE}

Algorithm:
  procedure EXPAND
    begin
      EXPANDING : = TRUE
      get First line of macro definition {prototyp}
      from DEFTAB
      set up arguments from macro invocation in
      ARGTAB
      write macro invocation to expanded file as a
      comment
      while not end of macro definition do
         begin
           GETLINE
           PROCESSLINE
           end {while}
      EXPANDING : = FALSE
    end {EXPAND}

procedure GETLINE
    begin
       if EXPANDING then.
```



```
       begin get next line of macro definition from DEFTAB
         substitute arguments from ARGTAB for positional
         notation
        end {if}
 else
   read next line from input file
end {GETLINE}
```

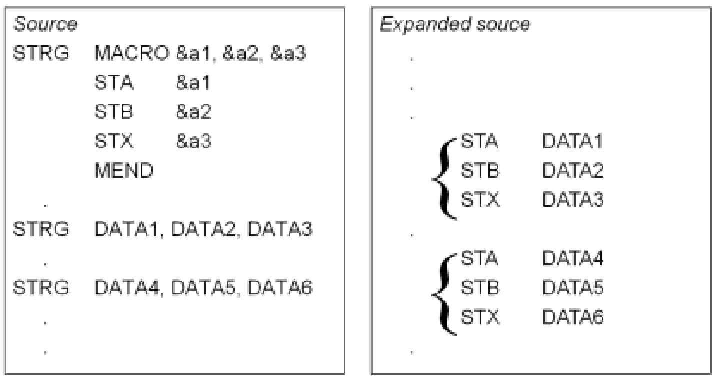
Conclusion:
Thus pass-I of Macro processor is implemented and MNT, MDT, & ALA file is generated.

Program:

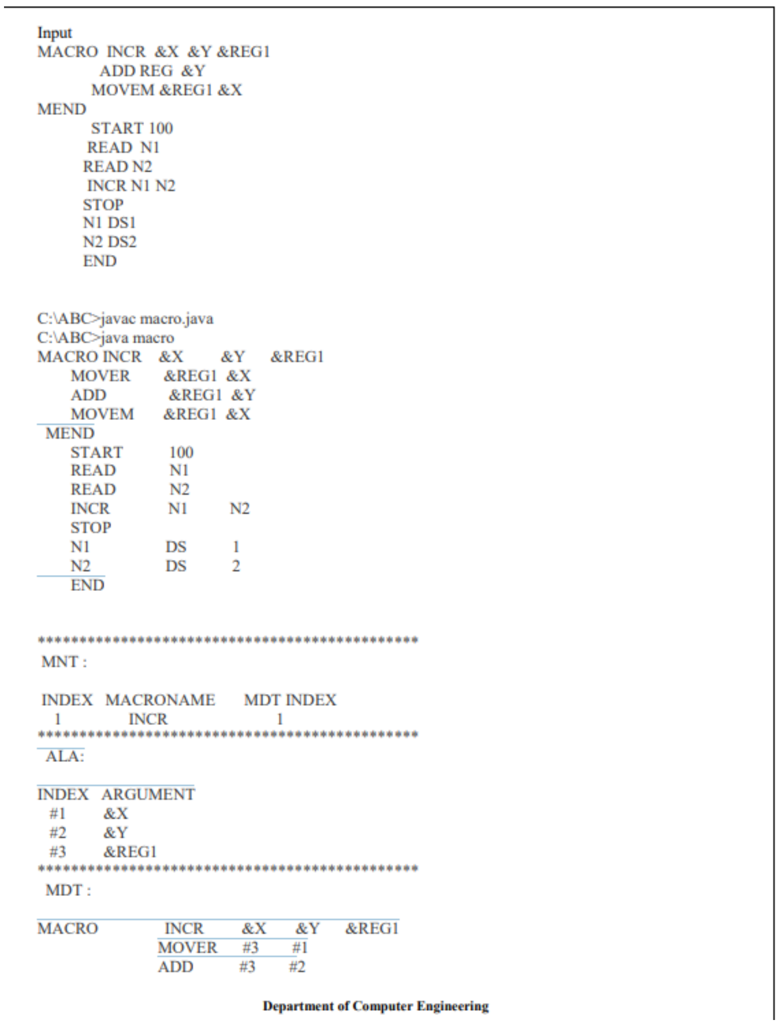


Example

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



```
Source                                  Expanded souce
STRG   MACRO &a1, &a2, &a3                .
       STA    &a1                         .
       STB    &a2                         .
       STX    &a3                       { STA    DATA1
       MEND                             { STB    DATA2
   .                                    { STX    DATA3
STRG   DATA1, DATA2, DATA3                .
   .                                    { STA    DATA4
STRG   DATA4, DATA5, DATA6              { STB    DATA5
   .                                    { STX    DATA6
   .                                      .
```



```
Input
MACRO  INCR  &X  &Y &REG1
          ADD REG  &Y
        MOVEM &REG1 &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          DS       2
     END


*************************************************
 MNT :

 INDEX  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
```

Department of Computer Engineering