Assignment No. 2: Design suitable data structures and implement Pass-I and Pass-II of a two-pass macro processor. The output of Pass-I (MNT, MDT and intermediate code file without any macro definitions) should be input for Pass-II.

Theory:

- **Macro processor.**

Macro processor is the system program which performs macro expansion i.e. replacement of macro call by corresponding set of instructions.
Therefore macro processor has to perform following main jobs:

1. Identification of macro definitions.
2. Storage of all macro definitions.
3. Identification of macro calls.
4. Replacement of macro calls by corresponding macro definition.

- **2 Pass Macro Processor**
  - PASS-1 purpose

Examines every opcode and saves all macro definition in MDT and save a copy of input text on secondary storage minus macro definition.
It also prepares MNT.

- PASS-2 purpose

Examines every opcode and replaces each macro name with appropriate macro definition.

- Specification of all the databases with their format
- **Macros with Positional and Keyword arguments**

Positional Arguments: Arguments are matched with dummy argument according to the order in which they appear.
i.e 'INCR A,B,C'
A, B & C replace the first second and third dummy argument.
Keyword argument: Allows reference to dummy argument by name as well as by position.
e.g. MAC3 &ARG1=DATA1, &ARG2=DATA2, &ARG3=DATA3

- **Conditional Macro Expansion:**

1. Conditional macros are the one, which will not get expanded to the same group of statements all the time but rather it depends on the kind of condition being put.
2. This allows conditional solution of machine instruction that appears in expansion of a macro call. The group of statements being replaced may vary in length and sequence, depending on the condition in the macro call.
3. Two macro processor pseudo opcodes, AIF & AGO are used.
4. AIF: It is a conditional branch pseudo opcode. It performs an arithmetic test and branches only if tested condition is true.
5. AGO: It is an unconditional branch pseudo-opcode or 'goto statement'.
   It specifies a label appearing on some other statement in the macro instruction definition.
6. These statements are directive to the macro processor and do not appear in macro Expansion.

Implementation Logic:

Input: A program written in assembly language containing macros with
arguments and condition

Output (Expected):

The above assembly program is to be stored separately in text file. The output expected is the expanded source program, i.e. expanded macro calls.

Expected outputs are:

- The expanded source program
- Macro Definition Table (MDT).
- The Macro Name Table (MNT).
- The updated Argument List Array (ALA).

Algorithm for Pass-I:

```
1. /* Initialization of counters for MDT and MNT*/
        MDTC=MNTC=1;
2.Read next instruction (and divide it into its various field as label , mnemonic opcode ,
Arguments)
                  i)      /*Check for macro definition start*/
                            if opcode=MACRO goto Step 5
                              else /* this is not macro definition*/
                                  go to step 4.
                  ii)     (A) Write copy of instruction to output of PASS-I
          (a)  Check whether opcode =END or not
          (b)  If OPCODE != END goto Step 2
          (c)  If OPCODE = END goto to Pass II i.e. End if this algorithm for pass I
                            B)  /* Start of macro definition is identified. Now Pass I will
                                process contents of macro definition after pseudo of
                                MACRO to MEND */
             a) Read next instruction
           (definitely this is a macro name instruction therefore as a processing of this
          instruction an entry will be made in MNT. ALA will be prepared for this macro,
          this macro name instruction will be entered in MDT*/
             b)Enter <macro-name MDTC> into MNT at MNTC
           /* current available rows in MDT and MNT are MDTC and MNTC, so macro
          name and its starting MDT index i.e. current value of MDTC is entered in MNT at
          available row i.e. MNTC*./
             c) MNTC=MNTC+1 /* To point next available row in MNT*/
             d)Prepare Argument  List Array
                /* ALA is partially constructed by Pass –I to assign universal integer index
          to dummy arguments*/
             e) Enter macroname instruction in MDT at MDTC.
             f) MDTC=MDTC+1.
 3. /* Process other instruction in macro definition inducing MEND instruction*/
          a.   Read next card
          b.   Substitute Index notations for dummy arguments.
          c.   Enter this instruction (where dummy arguments are replaced by integer
               indices) into MDT.
d.             MDTC:= MDTC +1
          e.   If OPCODE of this instruction is MEND then goto Step 2. else goto Step 6.a.
```

Algorithm for PASS II:

1. Read next instruction from source program outted by Pass-I and divide it into fields as label, mnemonic opcode, arguments.
2. Search through MNT to find match for OPCODE of instruction read in Step1 with macronames in MNT.
3. /* If no Macro call found*/

   if no match then it indicates that this instruction is not a macro call instruction and hence

   (a) Write this instruction to expanded source program file
   (b) Check whether OPCODE of this instruction is END or not.
   (c) If not END then it indicates that this is not end of source program and hence go to step 1
   (d) If OPCODE=END then this indicates end of source program and hence give the output of Pass II i.e. Expanded source program to assembler (This is the end of Pass II of Macroprocessor)
4. /* if macro name found */

   If OPCODE of (instruction read in step 1)= any macro name of MNT then it indicate that this instruction is macrocall instruction and hence.

   a. Obtain corresponding MDT index and assign to MDTP.
   b. Set up ALA (for the association of Integer Indices and actual parameters)
   c. MDTP:=MDTP+1
   d. Get next instruction from MDT
   e. Substitute actual arguments instead of integer indices.
   f. If OPCODE of this instruction is not MEND then write this instruction (after replacing integer indices by actual arguments) to the expanded source program.
   g. If OPCODE of this instruction is MEND then goto Step 1 again.

In Brief steps:

- Recognizing the Macro Definition.
- Save the Macro name in the MNT, and the Definition in the MDT.
- Recognize the Macro call and
- Use of ALA
- Expand the macro call.

Conclusion: Hence we successfully designed suitable data structures and implemented Pass-I Pass-II of a Two-pass macroprocessor.