Name: Digvijay D. Jondhale
Roll No: PC-32
PRN: 1032201770
Panel: C

System Software and Compiler lab
Assignment-03

Aim: Design suitable data structure & implement pass 1 of Two Pass Macroprocessor.

Objective: Design suitable data structure & implement pass 1 of Two Pass Macroprocessor. Input should consist of a one macro definition and one macro call and few assembly language instructions

Theory:

Description of Macroprocessor:

A macroprocessor is a program that extends the capabilities of an assembly language by allowing the use of macros. Macros are essentially user-defined shorthand notations for sequences of assembly language instructions or other constructs. They are particularly useful for simplifying and abstracting repetitive or complex tasks in assembly programming. The microprocessor takes source code containing macros as input and expands these macros into their corresponding assembly code before the actual assembly process.

Data structures Required for a Two-Pass Macroprocessor:
A two-pass macroprocessor involves two separate passes over the source code. During Pass I, it collects information about macros and their definitions, while Pass II expands the macros and produces the final assembly code. To achieve this, several data structures are required:

a] Macro Definition Table (MDT): This table stores the macro definitions encountered during Pass I. Each entry contains the macro name and its corresponding macro code.

b) Macro Invocation Table [MIT]: The MIT keeps track of macro invocations and their associated arguments during Pass I. It records the macro name and the arguments passed to it.

c) Argument Replacement Table (ART): The ART maps formal parameters of macros to the actual arguments provided during macro invocation. It is used during macro expansion in Pass II.

d) Local Symbol Table (LST): If macros introduce local symbols, the LST stores these symbols and their values within the scope of the macro. It helps maintain symbol uniqueness.

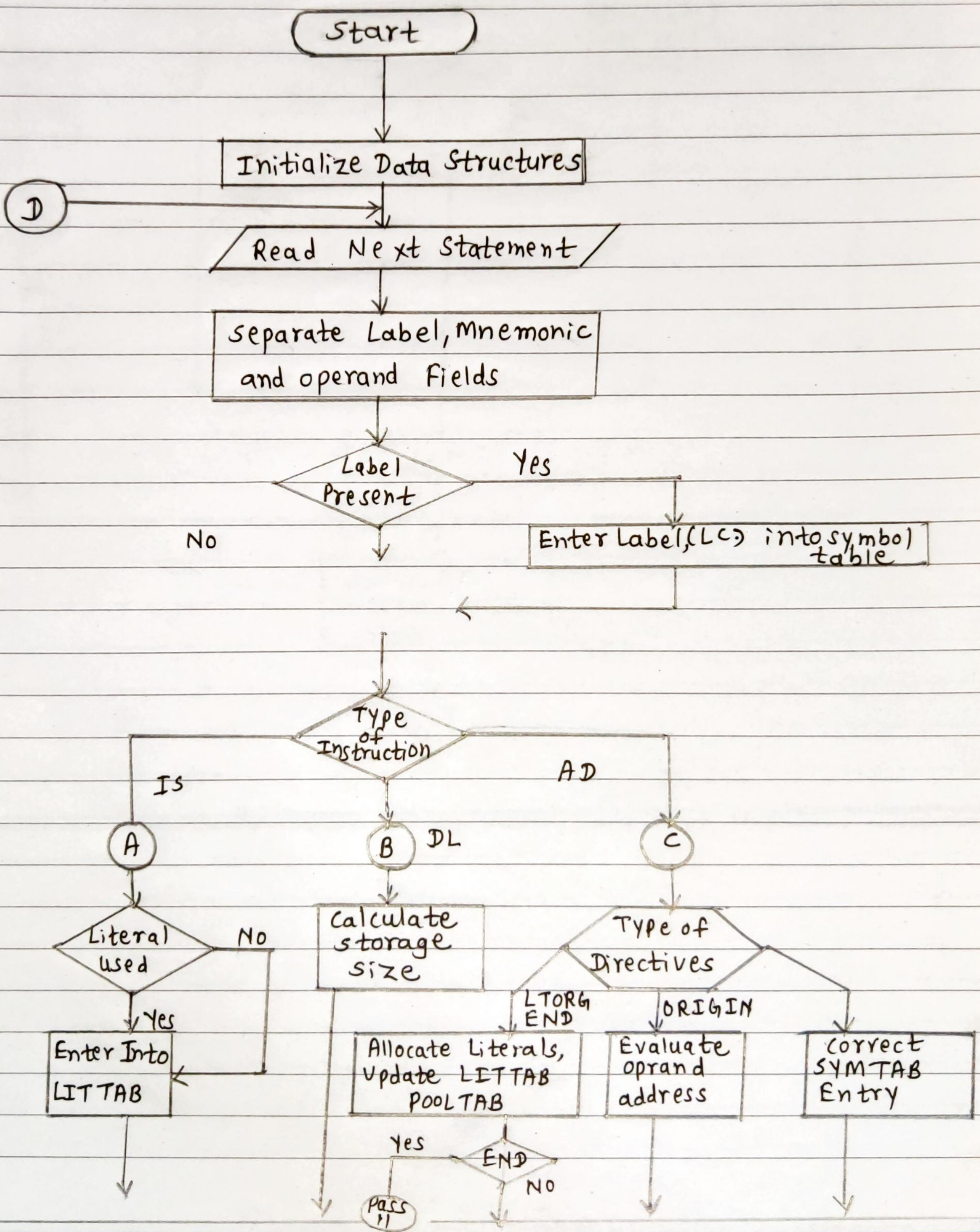
e) Global Symbol Table (GST): The GST stores global symbols of defined in the source code, allowing for cross-referencing between different macro invocations.

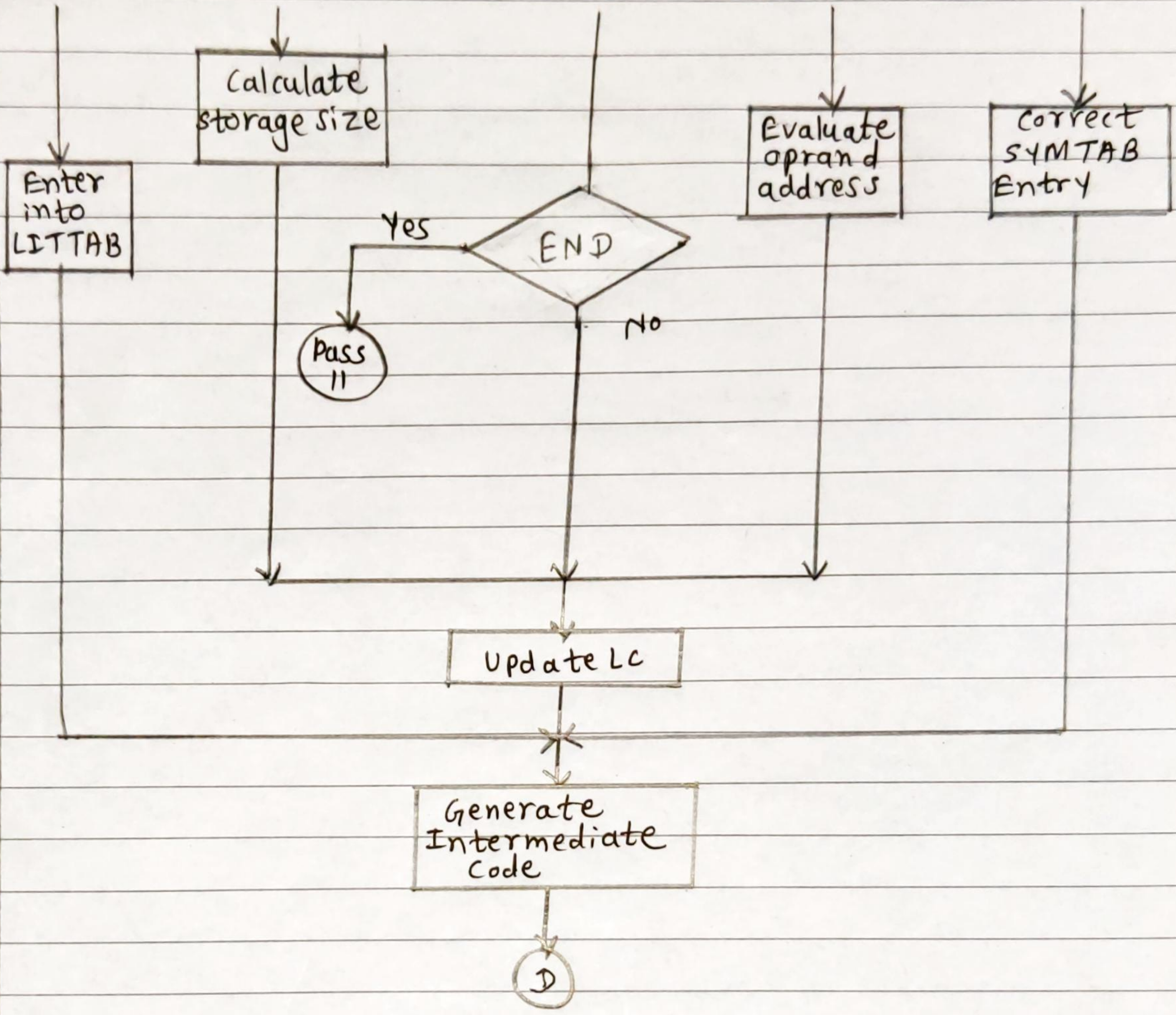
Algorithm for pass 1 :

Here's an algorithm that summarizes the steps involved in Pass 1 of a two-pass macroprocessor:

```
Initialize MDT, MIT, ART, LST, GST
Repeat until the end of source code:
Read the next line from the source code
 If the line is a macro definition :
 Parse the macro invocation and store it in MDT
 If the line is a macro invocation :
  Parse the macro invocation and store it in MIT
  Add the arguments to ART
  If the line contains global symbols:
   Process global symbols and update GST
```

Flowchart For Pass 1




Calculate storage size
Enter into LITTAB
Yes
END
No
Pass II
Evaluate oprand address
Correct SYMTAB Entry
Update LC
Generate Intermediate Code
D