(A)6.Write a program to generate MDT MNT(Macro Definition Table) of a two-pass Macro

processor.

INPUT/CODE

LOAD A

STORE B

MACRO ABC

LOAD p

SUB q

MEND

MACRO ADD1 ARG

LOAD X

STORE ARG

MEND

MACRO ADD5 A1, A2, A3

STORE A2

ADD1 5

ADD1 10

LOAD A1

LOAD A3

MEND

ABC

ADD5 D1, D2, D3

END

assembly\_code = [

"LOAD J",

"STORE M",

"MACRO EST",

"...continued...",

"LOAD P",

"ADD V",

"MACRO ADD7 P4, P5, P6",

"LOAD P5",

"LOAD e",

"ADD d",

"MEND",

"LOAD S",

"MACRO SUB4 ABC",

"LOAD U",

"STORE ABC",

"MEND",

"SUB4 XYZ",

"SUB 8",

"SUB 2",

"STORE P4",

"STORE P6",

"MEND",

"EST",

"ADD7 C4, C5, C6",

"SUB4 z",

"END",

]

# Step 1: Parse Macro Definitions

macro\_table = {}

current\_macro\_name = None

current\_macro\_body = []

for line in assembly\_code:

if line.startswith("MACRO"):

current\_macro\_name = line.split()[1]

current\_macro\_body = []

elif line.startswith("MEND"):

if current\_macro\_name:

macro\_table[current\_macro\_name] = current\_macro\_body.copy()

current\_macro\_name = None

elif current\_macro\_name:

current\_macro\_body.append(line)

# Display Macro Table

print("Macro Table:")

for macro\_name, macro\_body in macro\_table.items():

print(f"Macro Name: {macro\_name}")

print("Macro Body:")

for line in macro\_body:

print(f" {line}")

print() # Add an empty line for clarity between macros

# Step 2: Generate Intermediate Code with Macro Expansion

intermediate\_code = []

for line in assembly\_code:

if line.startswith("MACRO") or line.startswith("MEND"):

continue

parts = line.split()

if parts[0] in macro\_table:

macro\_body = macro\_table[parts[0]]

macro\_arguments = parts[1:]

for macro\_line in macro\_body:

expanded\_line = macro\_line

for idx, arg in enumerate(macro\_arguments):

parameter = f"A{idx + 1}" # Macro parameters are A1, A2, A3, ...

expanded\_line = expanded\_line.replace(parameter, arg)

intermediate\_code.append(expanded\_line)

else:

intermediate\_code.append(line)

# Step 3: Display Intermediate Code

print("Intermediate Code:")

for code\_line in intermediate\_code:

print(code\_line)