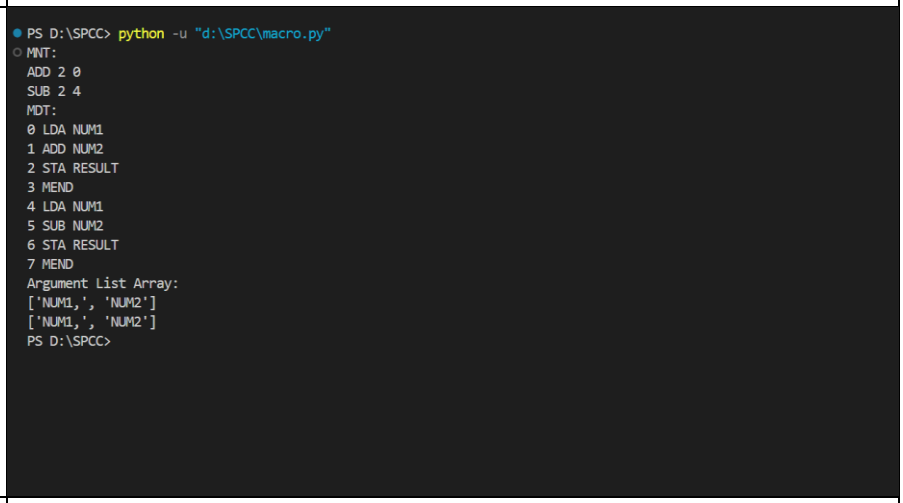
Experiment No.	**04**
Experiment Title	**Design and Implement Pass 1 of a Macro processor** **Task: Write a program to read a given text file which can identify the macros in the program and display MNT, MDT and Argument List Array.**
Student Name	Ronak Surve
Roll No.	64
Objectives:	• **Students will be able to learn and identify macro definition and macro name in a given program.** • **Students will be able to understand the use of Macros in a program**
Theory /Algorithm:	In a Macro processor, Pass 1 is responsible for building the Macro definition table and the Macro name table. The Macro definition table (MDT) contains the expanded Macro instructions, while the Macro name table (MNT) contains the Macro names, their argument lists, and the location of the corresponding Macro definition in the MDT. The following algorithm describes the theory behind Pass 1 of a Macro processor: Initialize the MDT and MNT. Loop through each line in the source code file. If the line contains a Macro definition, extract the Macro name, the number of arguments, and the argument list. Add the Macro name, the number of arguments, and the index of the first line of the Macro definition in the MDT to the MNT. Add the Macro instructions to the MDT, substituting the argument names with placeholders. If the line contains a Macro call, add the Macro call to the source code as it is, and add the argument values to the Argument List Array. If the line does not contain a Macro definition or a Macro call, add it to the source code as it is. End the loop. Output the MDT, MNT, and the Argument List Array. In Pass 1, the Macro processor scans the source code for Macro definitions and Macro calls. When it finds a Macro definition, it extracts the Macro name, the number of arguments, and the argument list. It adds the Macro name, the number of arguments, and the index of the first line of the Macro definition in the MDT to the MNT. It then adds the Macro instructions to the MDT, substituting the argument names with placeholders.

	When it finds a Macro call, it adds the Macro call to the source code as it is and adds the argument values to the Argument List Array. The output of Pass 1 includes the MDT, MNT, and the Argument List Array. The MDT contains the expanded Macro instructions, while the MNT contains the Macro names, their argument lists, and the location of the corresponding Macro definition in the MDT. The Argument List Array contains the values of the Macro arguments. Pass 1 is important because it sets up the data structures required for Pass 2 to perform the Macro substitution.

Program Code:	# open the input file for reading
	with open('input_file.txt', 'r') as file:
	
	  # initialize the Macro Name Table (MNT)
	  mnt = {}
	
	  # initialize the Macro Definition Table (MDT)
	  mdt = []
	
	  # initialize the Argument List Array
	  arg_list_arr = []
	
	  # initialize a flag to indicate whether we are inside a Macro definition
	  inside_macro = False
	
	  # loop through each line in the file
	  for line in file:
	
	  # check if the line starts with a Macro definition
	  if line.startswith('MACRO'):
	
	  # extract the Macro name and argument list
	  macro_name, arg_list = line.split()[1], line.split()[2:]
	
	  # add the Macro name and argument list to the MNT
	  mnt[macro_name] = (len(arg_list), len(mdt))
	
	  # add the argument list to the Argument List Array
	  arg_list_arr.append(arg_list)
	
	  # set the flag to indicate that we are inside a Macro definition
	  inside_macro = True
	
	  # check if we are inside a Macro definition
	  elif inside_macro:
	
	  # add the line to the MDT
	  mdt.append(line.strip())
	
	  # check if the line ends with 'MEND'
	  if line.endswith('MEND\n'):
	
	  # set the flag to indicate that we are no longer inside a Macro definition
	  inside_macro = False
	
	  # print the Macro Name Table (MNT)
	  print('MNT:')
	  for macro_name, (num_args, mdt_index) in mnt.items():
	  print(f'{macro_name} {num_args} {mdt_index}')

	# print the Macro Definition Table (MDT)
	print('MDT:')
	for i, macro_definition in enumerate(mdt):
	print(f'{i} {macro_definition}')
	
	# print the Argument List Array
	print('Argument List Array:')
	for arg_list in arg_list_arr:
	print(arg_list)

Input to the Program:	START 100
	MACRO ADD NUM1, NUM2
	LDA NUM1
	ADD NUM2
	STA RESULT
	MEND
	MACRO SUB NUM1, NUM2
	LDA NUM1
	SUB NUM2
	STA RESULT
	MEND
	ADD 10, 20
	SUB 50, 30
	END

Output of the program:	
Outcome of the Experiment:	Successfully implemented a program to identify macro definition and its implementation.
References:	https://forgetcode.com/c/103-pass-one-of-a-two-pass-assembler https://www.geeksforgeeks.org/introduction-of-assembler/