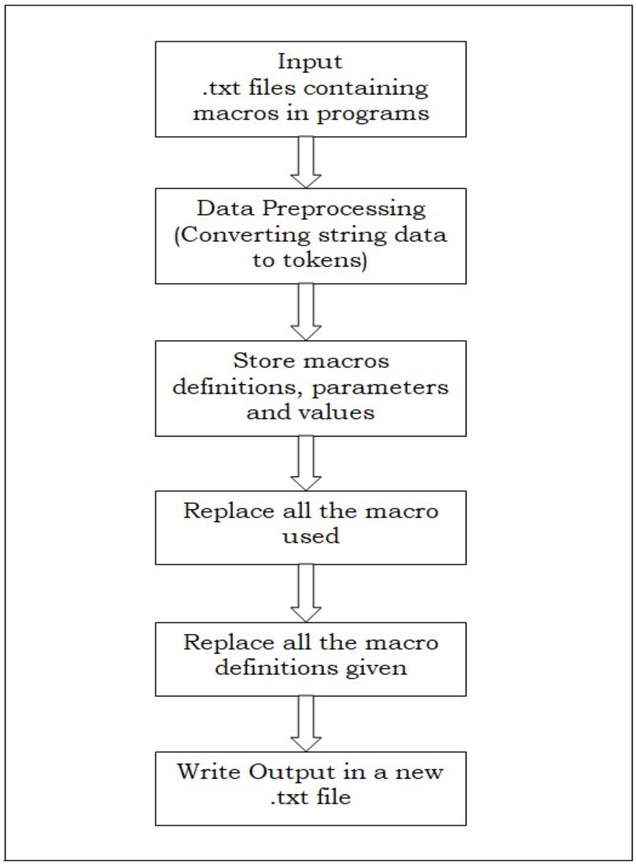
DAYANANDA SAGAR COLLEGE OF ENGINEERING
COMPUTER SCIENCE & ENGINEERING

Minor Project- Report
Apr 2021-Jul 2021

Course Faculty: Prof. Asnika S Course Name & code: System Software (18CS6DCSSW)
Semester: 6 Date: 09/05/2021

TITLE OF THE PROJECT	Macro Preprocessor Implementation			
STUDENT NAME	Kumar Gaurav	Mudimelavikas Reddy	Nikhil Raj	Nikita
USN	1DS18CS170	1DS18CS171	1DS18CS172	1DS18CS173
INDIVIDUAL CONTRIBUTION	Worked on Data Structures to stores macros parameters and values. Also worked on logic to store and replace macro in output files.	Worked on the data preprocessing of the input files to convert the strings of data into tokens.	Worked on Data Structures to stores macros parameters and values. Also worked on logic to store and replace macro in output files.	Worked on the data preprocessing of the input files to convert the strings of data into tokens.
GUIDE	Prof. Navyashree T M			
PROJECT ABSTRACT :	Macro Preprocessors are used by most of the programming languages. Most macro preprocessor, are either syntactically tied to the language they support (e.g The C Preprocessor) or they are limited in their functionality (e.g in C++). A macro (which stands for "macroinstruction") is used to make certain tasks less repetitive by representing a complicated sequence of commands or statements into a shorthand notation. A Macro Preprocessor inputs a program with macro definitions and calls and outputs program without macros for compilation. It replaces each macro invocation (call) with the corresponding sequence of statements (expansion). In this project, we are implementing a Macro Preprocessor based on Python scripting language. It is not integrated with a particular language or piece of software and it is suitable for both low level language (like NASM) and high level languages (like Python and C). Our Macro Preprocessor will input a piece of code containing macros as .txt files and output the code without macros in the same format. We will include the three main functions of macro definition, macro expansion, and file inclusion. This will be useful for macro based generation of files.			
PLATFORM USED (H/W & S/W TOOLS TO BE USED	Python, Visual Studio code			

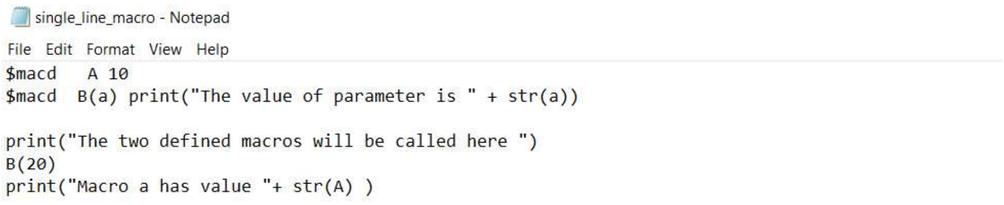
INTRODUCTION	The Macro Preprocessor application inputs a program with macro definitions and calls and outputs program without macros for compilation. The main program does 3 tasks mainly: • It takes the input file containing the macros definitions. • Stores the different macros definitions. • Removes the macros definitions and stores the result in output file. We've created different input files for different conditions like single-line macros, multi-line macros, nested macros etc. so the main program first reads these input files from their .txt format and then they are passed on to be converted into tokens. The tokens created are later used for building further logic of single/multi/nested macros. The logic defining all the macros definitions are written through which the input file is parsed. The logic for each of these macros and tokens is written in a different python file and called in main program. Once it is done, the program now starts removing the macros definitions and then creates another text file where it stores the result. The result thus formed would be the one without any macros. So these files can be said as preprocessed before the compilers begin to compile the program. Since the macros is already preprocessed and removed, it makes the compiling much faster and efficient.
DESIGN	 Input .txt files containing macros in programs Data Preprocessing (Converting string data to tokens) Store macros definitions, parameters and values Replace all the macro used Replace all the macro definitions given Write Output in a new .txt file

DAYANANDA SAGAR COLLEGE OF ENGINEERING
COMPUTER SCIENCE & ENGINEERING

PROJECT SOURCE CODE LINK (GITHUB/ GOOGLE DRIVE)	https://drive.google.com/drive/folders/1uma84oyrRzX_fpQJwCgNB-pUJUVxOJdn?usp=sharing
CONCLUSION /FUTURE ENHANCEMENT	The application implements a macro preprocessor for both high and low level programming languages. While the application replaces each macro invocation with the corresponding sequence of statements for single line, multiline and nested macros, some further enhancement can be done in the application to make it work with conditional macros as well. Apart from that, a well designed UI can also be implemented to work with the application efficiently.

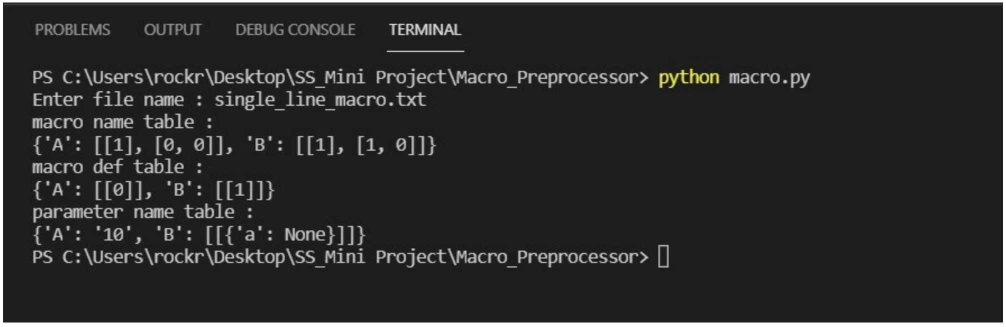
UI SCREENSHOTS

Example 1 – Single Line Macro



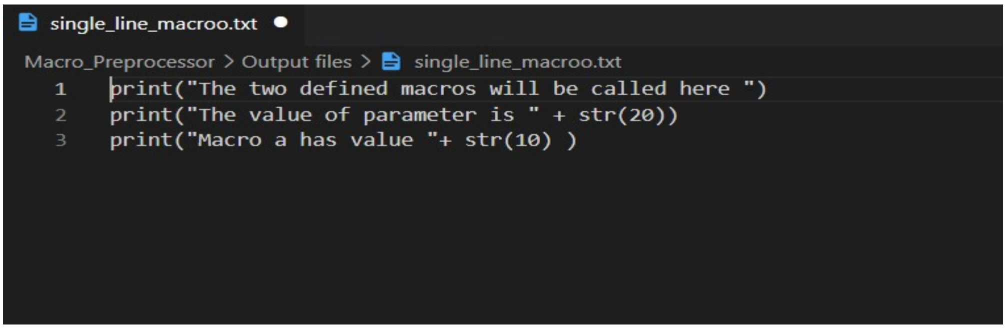
```
$macd   A 10
$macd  B(a) print("The value of parameter is " + str(a))

print("The two defined macros will be called here ")
B(20)
print("Macro a has value "+ str(A) )
```

Input File



```
PS C:\Users\rockr\Desktop\SS_Mini Project\Macro_Preprocessor> python macro.py
Enter file name : single_line_macro.txt
macro name table :
{'A': [[1], [0, 0]], 'B': [[1], [1, 0]]}
macro def table :
{'A': [[0]], 'B': [[1]]}
parameter name table :
{'A': '10', 'B': [[{'a': None}]]}
PS C:\Users\rockr\Desktop\SS_Mini Project\Macro_Preprocessor>
```

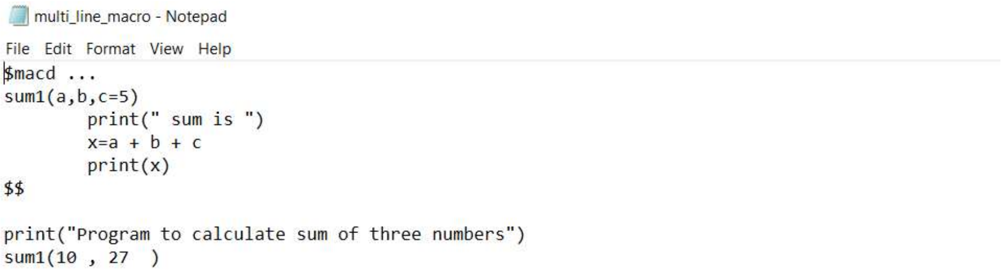
Console Output



```
print("The two defined macros will be called here ")
print("The value of parameter is " + str(20))
print("Macro a has value "+ str(10) )
```

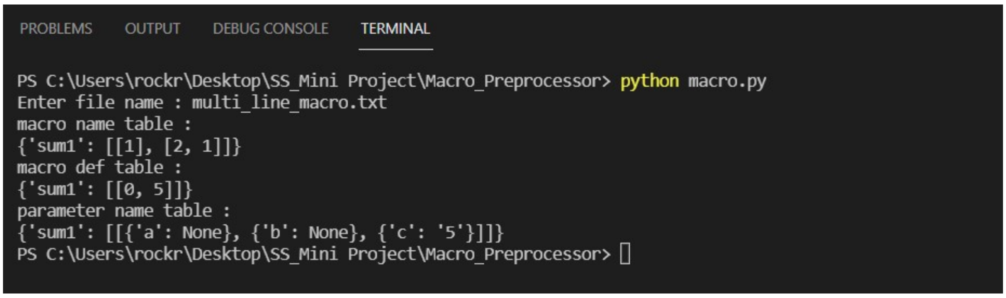
Output file

Example 2 – Multi-line Macro



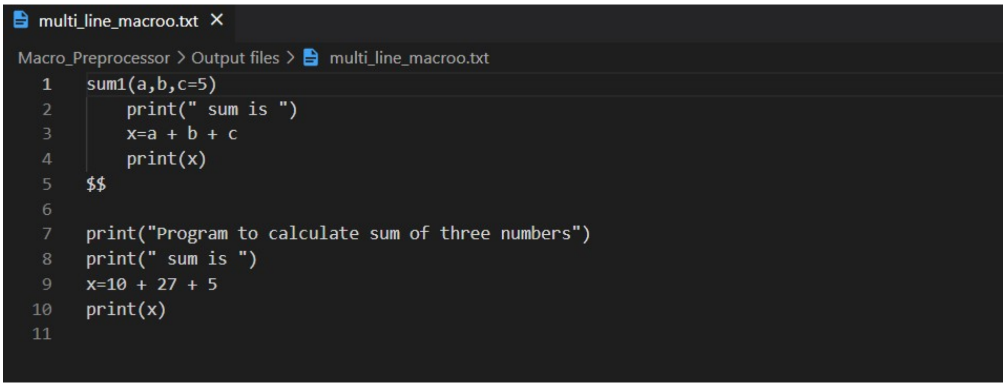
```
$macd ...
sum1(a,b,c=5)
        print(" sum is ")
        x=a + b + c
        print(x)
$$

print("Program to calculate sum of three numbers")
sum1(10 , 27  )
```

Input file



```
PS C:\Users\rockr\Desktop\SS_Mini Project\Macro_Preprocessor> python macro.py
Enter file name : multi_line_macro.txt
macro name table :
{'sum1': [[1], [2, 1]]}
macro def table :
{'sum1': [[0, 5]]}
parameter name table :
{'sum1': [[{'a': None}, {'b': None}, {'c': '5'}]]}
PS C:\Users\rockr\Desktop\SS_Mini Project\Macro_Preprocessor>
```

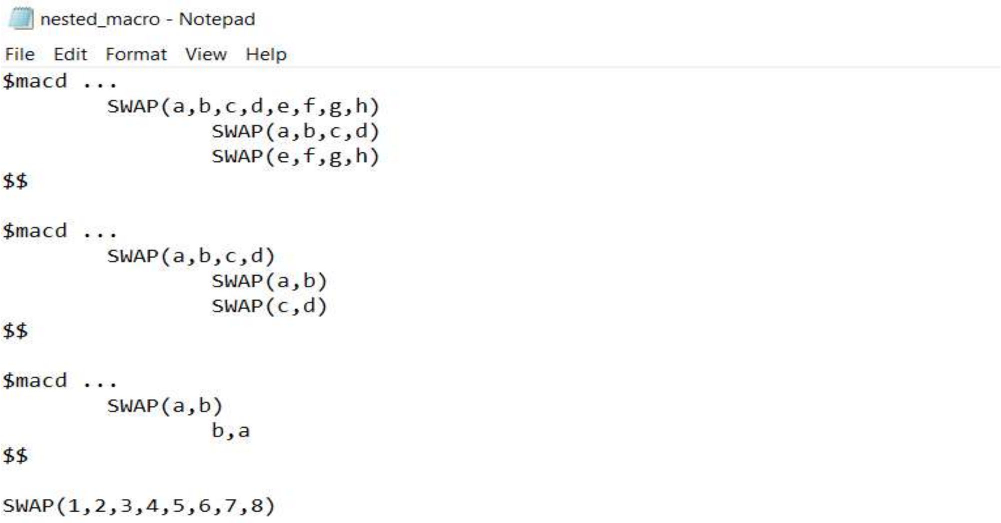
Console Output



```
sum1(a,b,c=5)
    print(" sum is ")
    x=a + b + c
    print(x)
$$

print("Program to calculate sum of three numbers")
print(" sum is ")
x=10 + 27 + 5
print(x)
```

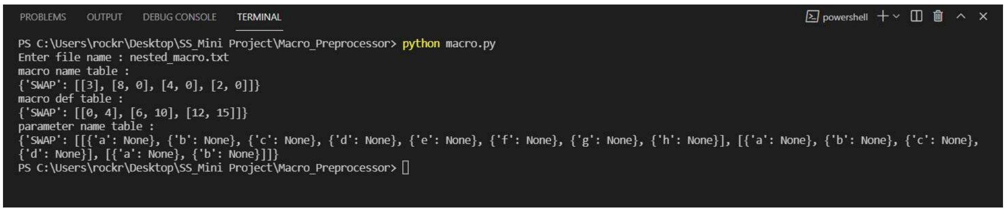
Output file

Example 3 – Nested macro



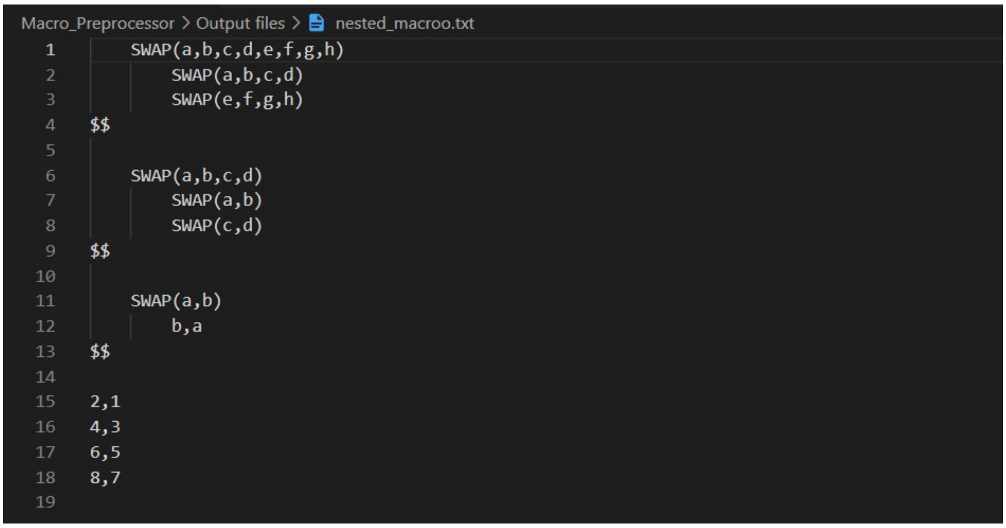
```
$macd ...
        SWAP(a,b,c,d,e,f,g,h)
                SWAP(a,b,c,d)
                SWAP(e,f,g,h)
$$

$macd ...
        SWAP(a,b,c,d)
                SWAP(a,b)
                SWAP(c,d)
$$

$macd ...
        SWAP(a,b)
                b,a
$$

SWAP(1,2,3,4,5,6,7,8)
```

Input



```
PS C:\Users\rockr\Desktop\SS_Mini Project\Macro_Preprocessor> python macro.py
Enter file name : nested_macro.txt
macro name table :
{'SWAP': [[3], [8, 0], [4, 0], [2, 0]]}
macro def table :
{'SWAP': [[0, 4], [6, 10], [12, 15]]}
parameter name table :
{'SWAP': [[{'a': None}, {'b': None}, {'c': None}, {'d': None}, {'e': None}, {'f': None}, {'g': None}, {'h': None}], [{'a': None}, {'b': None}, {'c': None},
{'d': None}], [{'a': None}, {'b': None}]]}
PS C:\Users\rockr\Desktop\SS_Mini Project\Macro_Preprocessor> 
```

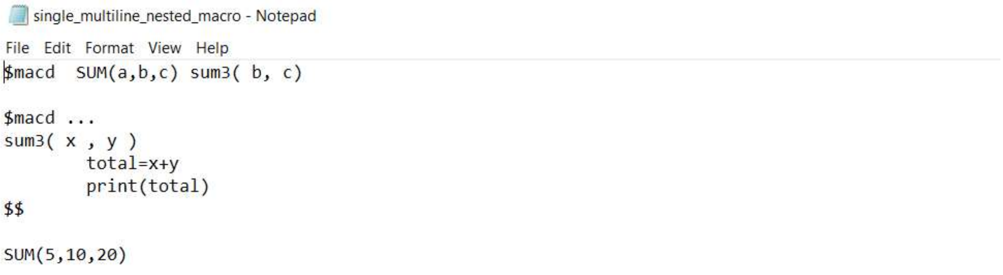
Console Output



```
    SWAP(a,b,c,d,e,f,g,h)
        SWAP(a,b,c,d)
        SWAP(e,f,g,h)
$$

    SWAP(a,b,c,d)
        SWAP(a,b)
        SWAP(c,d)
$$

    SWAP(a,b)
        b,a
$$

2,1
4,3
6,5
8,7

```

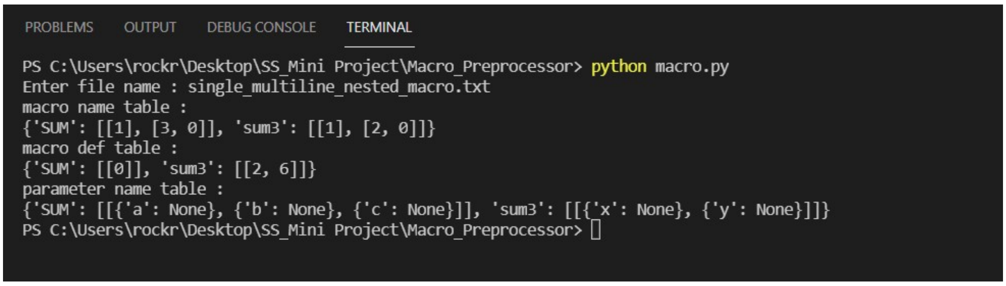
Output file

Example 4 – Multi line nested macro


single_multiline_nested_macro - Notepad
File Edit Format View Help

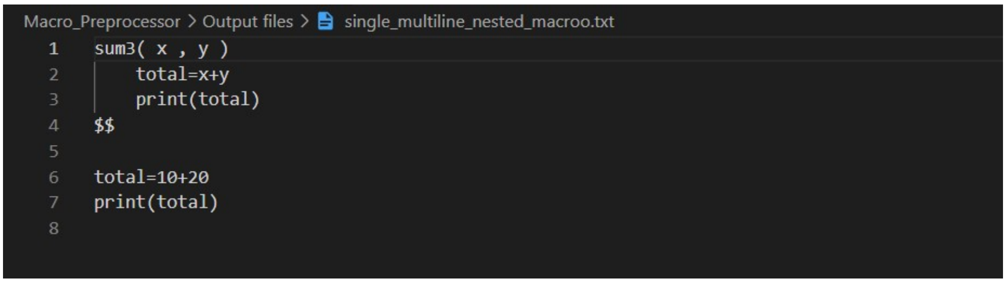
```
$macd  SUM(a,b,c) sum3( b, c)

$macd ...
sum3( x , y )
        total=x+y
        print(total)
$$

SUM(5,10,20)
```

Input



```
PS C:\Users\rockr\Desktop\SS_Mini Project\Macro_Preprocessor> python macro.py
Enter file name : single_multiline_nested_macro.txt
macro name table :
{'SUM': [[1], [3, 0]], 'sum3': [[1], [2, 0]]}
macro def table :
{'SUM': [[0]], 'sum3': [[2, 6]]}
parameter name table :
{'SUM': [[{'a': None}, {'b': None}, {'c': None}]], 'sum3': [[{'x': None}, {'y': None}]]}
PS C:\Users\rockr\Desktop\SS_Mini Project\Macro_Preprocessor> 
```

Console Output


Macro_Preprocessor > Output files > single_multiline_nested_macroo.txt

```
sum3( x , y )
    total=x+y
    print(total)
$$

total=10+20
print(total)

```

Output file