Experiment no 3

```
import re
# Regular expressions for macro definition and macro call
DEFINITION_REGEX = r'^\s*MACRO\s+(\w+)\s+(.*)$'
CALL_REGEX = r'(\w+)\((.*)\)'
class Macro:
    def __init__(self, name, parameters, code):
        self.name = name
        self.parameters = parameters
        self.code = code
class TwoPassMacroProcessor:
    def __init__(self):
        self.macros = {}
    def first_pass(self, source):
        # Extract macro definitions and build macro table
        lines = source.split('\n')
        i = 0
        while i < len(lines):
            match = re.match(DEFINITION_REGEX, lines[i])
            if match:
                name, parameters = match.groups()
                parameters = [p.strip() for p in parameters.split(',')]
                code = []
                i += 1
                while i < len(lines) and not re.match(DEFINITION_REGEX, lines[i]):
                    code.append(lines[i])
                    i += 1
                macro = Macro(name, parameters, code)
                self.macros[name] = macro
            else:
                i += 1
    def second_pass(self, source):
        # Replace macro calls with expanded code
        lines = source.split('\n')
        for i in range(len(lines)):
            match = re.search(CALL_REGEX, lines[i])
            if match:
                name, arguments = match.groups()
                arguments = [a.strip() for a in arguments.split(',')]
                macro = self.macros.get(name)
                if macro:
                    expanded_code = macro.code.copy()
                    for j in range(len(arguments)):
                        expanded_code = [re.sub(r'\b' + macro.parameters[j] + r'\b', arguments[j], line) for line
in expanded_code]
                    lines[i:i+1] = expanded_code
        return '\n'.join(lines)
    def process(self, source):
        self.first_pass(source)
        return self.second_pass(source)
source = '''
```

```
MACRO ADD(a, b)
   MOV AX, a
   ADD AX, b
   MOV result, AX
ENDM
ADD(10, 20)
'''
processor = TwoPassMacroProcessor()
result = processor.process(source)
print(result)
```

Output:

```
C:\Python3.10\python.exe C:\COLLEGE\SPCC\EXP-3.py

MACRO ADD(a, b)
    MOV AX, a
    ADD AX, b
    MOV result, AX
ENDM

ADD(10, 20)


Process finished with exit code 0
```