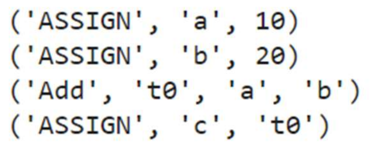
Experiment no 6

```
import ast
class IntermediateCodeGenerator(ast.NodeVisitor):
    def __init__(self):
        self.instructions = []
        self.temp_count = 0
    def new_temp(self):
        temp = f"t{self.temp_count}"
        self.temp_count += 1
        return temp
    def visit_Assign(self, node):
        target = node.targets[0].id
        value = self.visit(node.value)
        self.instructions.append(('ASSIGN', target, value))
    def visit_BinOp(self, node):
        left = self.visit(node.left)
        right = self.visit(node.right)
        op = node.op.__class__.__name__
        temp = self.new_temp()
        self.instructions.append((op, temp, left, right))
        return temp
    def visit_Num(self, node):
        return node.n
    def visit_Name(self, node):
        return node.id
    def visit_Print(self, node):
        value = self.visit(node.values[0])
        self.instructions.append(('PRINT', value))
def generate_intermediate_code(source_code):
    ast_tree = ast.parse(source_code)
    icg = IntermediateCodeGenerator()
    icg.visit(ast_tree)
    return icg.instructions
# Example usage
source_code = """
a = 10
b = 20
c = a + b
print(c)
"""
instructions = generate_intermediate_code(source_code)
for instruction in instructions:
    print(instruction)
```

Output :



```
('ASSIGN', 'a', 10)
('ASSIGN', 'b', 20)
('Add', 't0', 'a', 'b')
('ASSIGN', 'c', 't0')
```