Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Лабораторна робота №1

Дисципліна: Комп'ютерна дискретна математика

Виконав

Студент групи ПЗПІ-23-3

Харченко Федір Олександрович

Перевірив

Старший викладач каф. ПІ

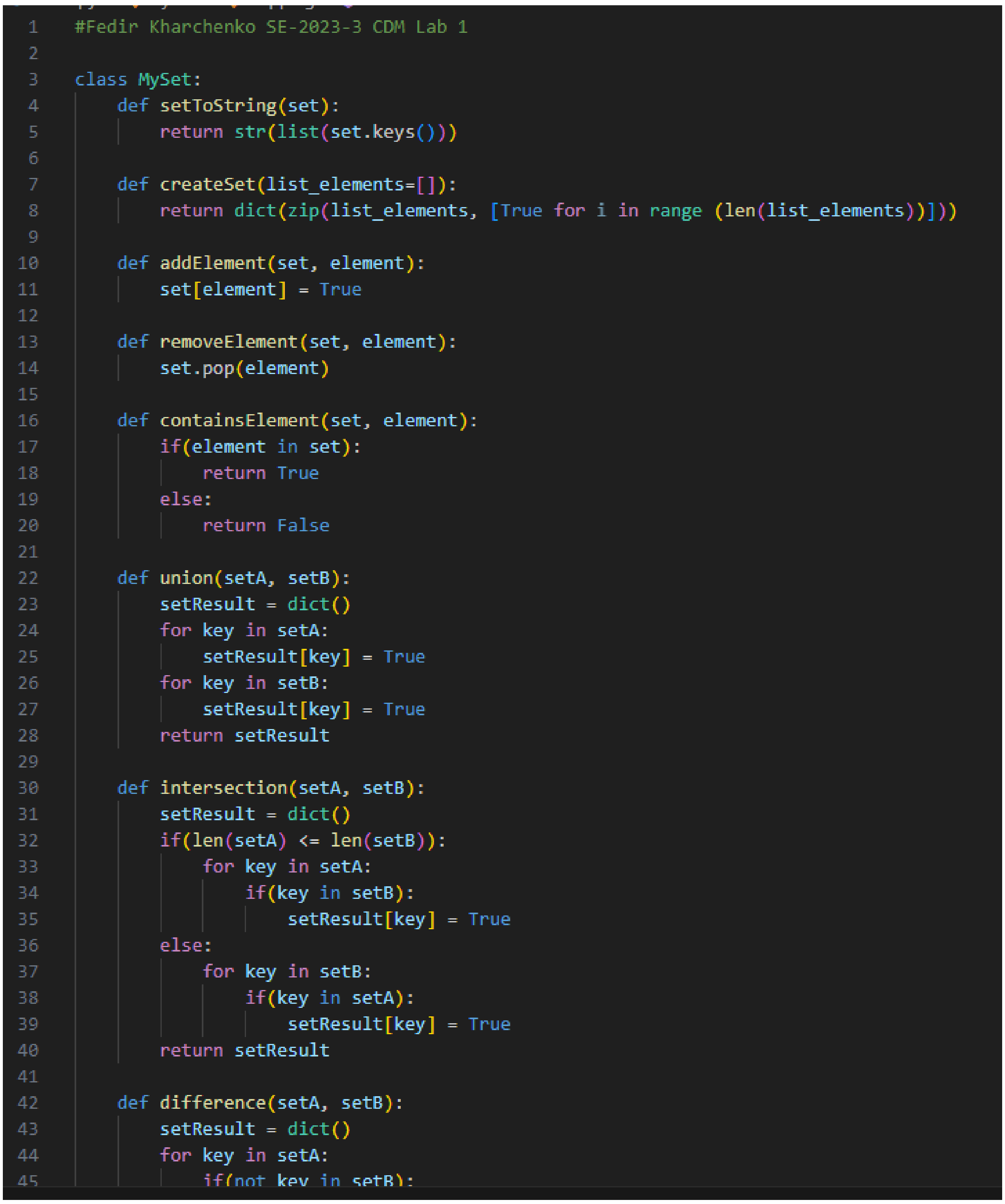
Терещенко Гліб Юрійович

2023

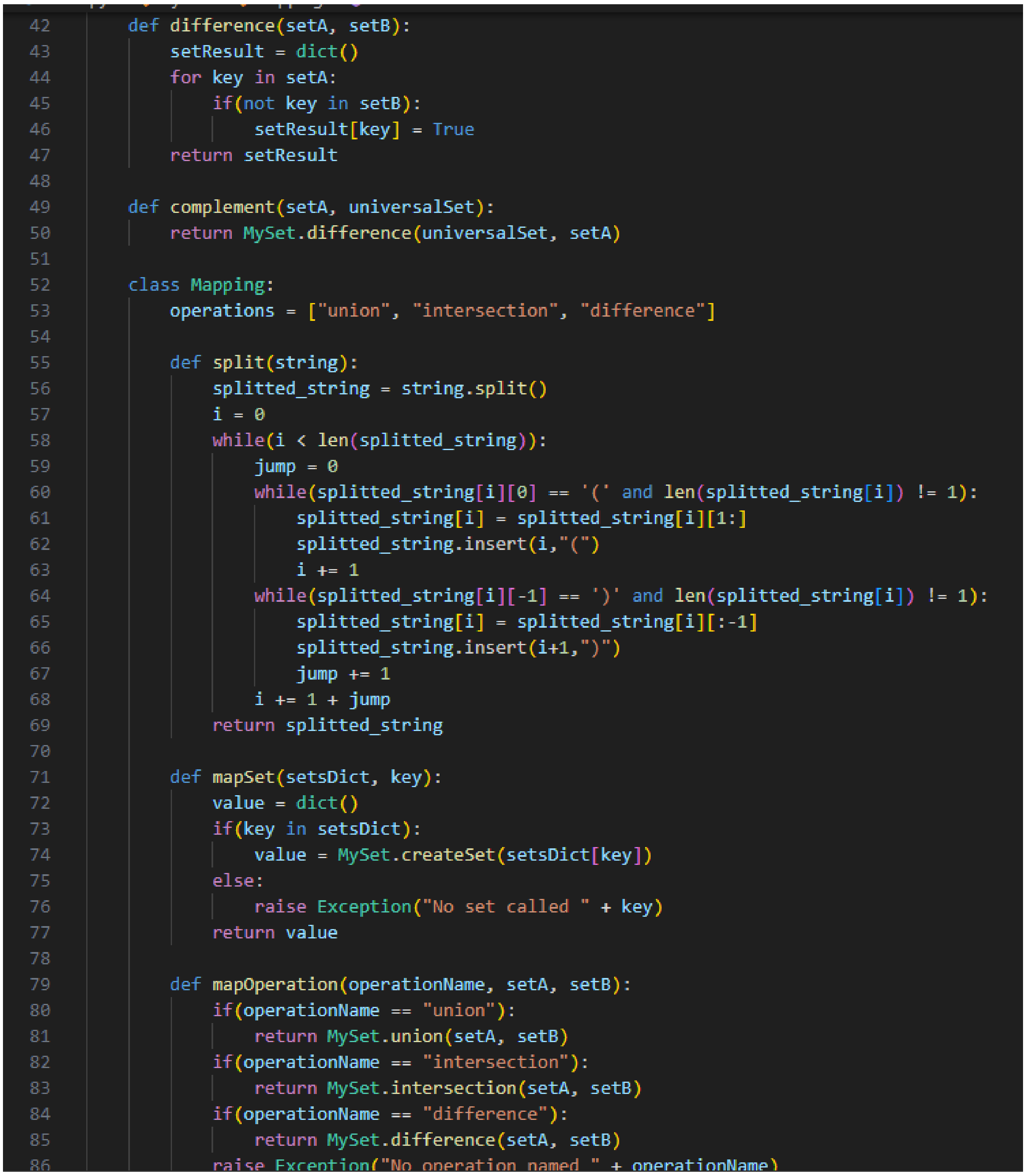
Тема: Basic Set Operations

Мета: Implement foundational set operations without using pre-built libraries or classes.

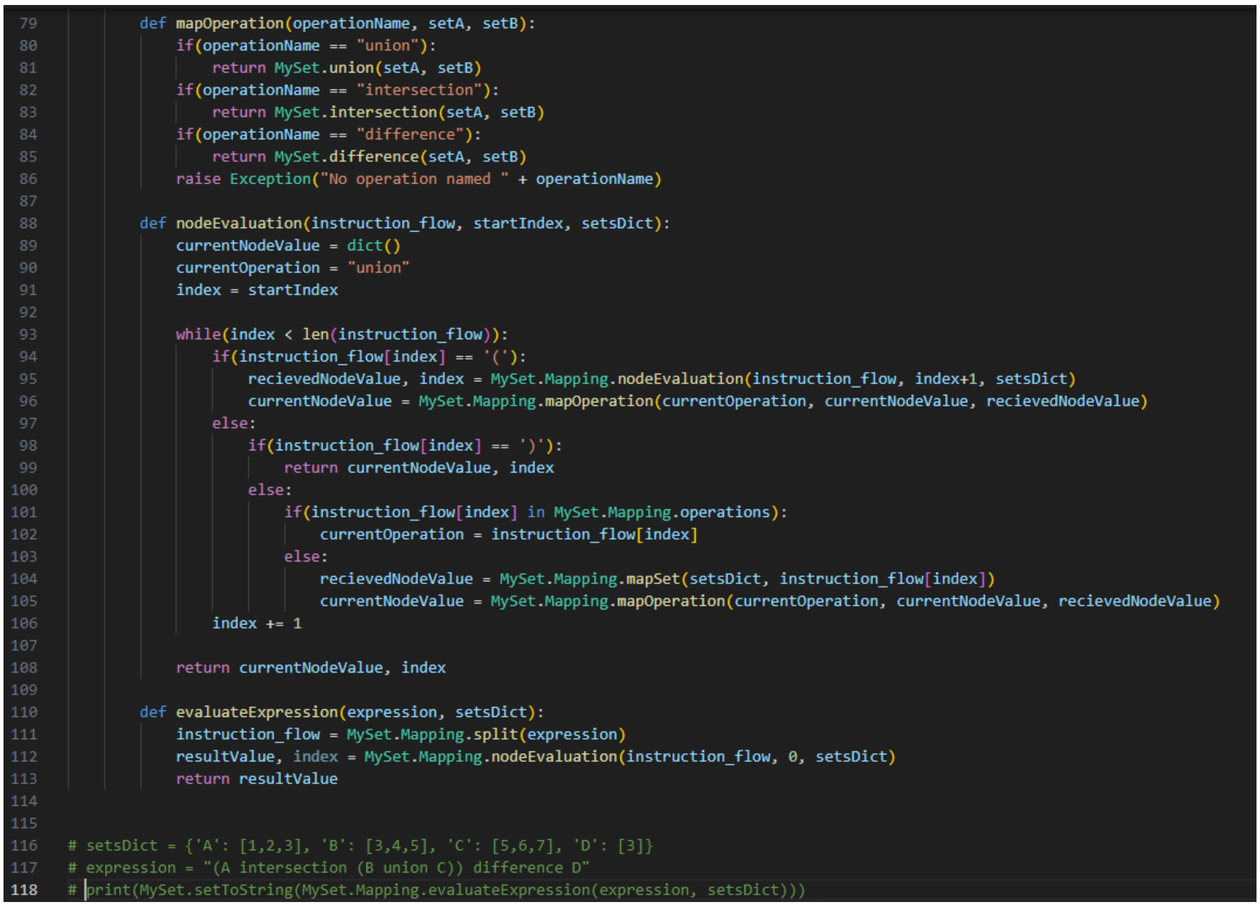
Код програми:



```python
#Fedir Kharchenko SE-2023-3 CDM Lab 1

class MySet:
    def setToString(set):
        return str(list(set.keys()))

    def createSet(list_elements=[]):
        return dict(zip(list_elements, [True for i in range (len(list_elements))]))

    def addElement(set, element):
        set[element] = True

    def removeElement(set, element):
        set.pop(element)

    def containsElement(set, element):
        if(element in set):
            return True
        else:
            return False

    def union(setA, setB):
        setResult = dict()
        for key in setA:
            setResult[key] = True
        for key in setB:
            setResult[key] = True
        return setResult

    def intersection(setA, setB):
        setResult = dict()
        if(len(setA) <= len(setB)):
            for key in setA:
                if(key in setB):
                    setResult[key] = True
        else:
            for key in setB:
                if(key in setA):
                    setResult[key] = True
        return setResult

    def difference(setA, setB):
        setResult = dict()
        for key in setA:
            if(not key in setB):
```



```python
    def difference(setA, setB):
        setResult = dict()
        for key in setA:
            if(not key in setB):
                setResult[key] = True
        return setResult

    def complement(setA, universalSet):
        return MySet.difference(universalSet, setA)

    class Mapping:
        operations = ["union", "intersection", "difference"]

        def split(string):
            splitted_string = string.split()
            i = 0
            while(i < len(splitted_string)):
                jump = 0
                while(splitted_string[i][0] == '(' and len(splitted_string[i]) != 1):
                    splitted_string[i] = splitted_string[i][1:]
                    splitted_string.insert(i,"(")
                    i += 1
                while(splitted_string[i][-1] == ')' and len(splitted_string[i]) != 1):
                    splitted_string[i] = splitted_string[i][:-1]
                    splitted_string.insert(i+1,")")
                    jump += 1
                i += 1 + jump
            return splitted_string

        def mapSet(setsDict, key):
            value = dict()
            if(key in setsDict):
                value = MySet.createSet(setsDict[key])
            else:
                raise Exception("No set called " + key)
            return value

        def mapOperation(operationName, setA, setB):
            if(operationName == "union"):
                return MySet.union(setA, setB)
            if(operationName == "intersection"):
                return MySet.intersection(setA, setB)
            if(operationName == "difference"):
                return MySet.difference(setA, setB)
            raise Exception("No operation named " + operationName)
```



```python
        def mapOperation(operationName, setA, setB):
            if(operationName == "union"):
                return MySet.union(setA, setB)
            if(operationName == "intersection"):
                return MySet.intersection(setA, setB)
            if(operationName == "difference"):
                return MySet.difference(setA, setB)
            raise Exception("No operation named " + operationName)

        def nodeEvaluation(instruction_flow, startIndex, setsDict):
            currentNodeValue = dict()
            currentOperation = "union"
            index = startIndex

            while(index < len(instruction_flow)):
                if(instruction_flow[index] == '('):
                    recievedNodeValue, index = MySet.Mapping.nodeEvaluation(instruction_flow, index+1, setsDict)
                    currentNodeValue = MySet.Mapping.mapOperation(currentOperation, currentNodeValue, recievedNodeValue)
                else:
                    if(instruction_flow[index] == ')'):
                        return currentNodeValue, index
                    else:
                        if(instruction_flow[index] in MySet.Mapping.operations):
                            currentOperation = instruction_flow[index]
                        else:
                            recievedNodeValue = MySet.Mapping.mapSet(setsDict, instruction_flow[index])
                            currentNodeValue = MySet.Mapping.mapOperation(currentOperation, currentNodeValue, recievedNodeValue)
                index += 1

            return currentNodeValue, index

        def evaluateExpression(expression, setsDict):
            instruction_flow = MySet.Mapping.split(expression)
            resultValue, index = MySet.Mapping.nodeEvaluation(instruction_flow, 0, setsDict)
            return resultValue


# setsDict = {'A': [1,2,3], 'B': [3,4,5], 'C': [5,6,7], 'D': [3]}
# expression = "(A intersection (B union C)) difference D"
# print(MySet.setToString(MySet.Mapping.evaluateExpression(expression, setsDict)))
```

Висновок: All basic set operation were implemented. Time complexity of all operations is as efficient as set’s one. Additionally, the equation evaluator was implemented. It supports all kinds of operations (union, intersection and difference) as well as computation order which can be specified by “(” and “)”.