

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

Тема: Basic Set Operations

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

Код програми:

<https://github.com/rintaro129/CDM/tree/main/LAB1>

```
1  #Fedir Kharchenko SE-2023-3 CDM Lab 1
2
3  class MySet:
4      def setToString(set):
5          |   return str(list(set.keys()))
6
7      def createSet(list_elements=[]):
8          |   return dict(zip(list_elements, [True for i in range (len(list_elements))]))
9
10     def addElement(set, element):
11         |   set[element] = True
12
13     def removeElement(set, element):
14         |   set.pop(element)
15
16     def containsElement(set, element):
17         |   if(element in set):
18             |       return True
19         |   else:
20             |       return False
21
22     def union(setA, setB):
23         |   setResult = dict()
24         |   for key in setA:
25             |       setResult[key] = True
26         |   for key in setB:
27             |       setResult[key] = True
28         |   return setResult
29
30     def intersection(setA, setB):
31         |   setResult = dict()
32         |   if(len(setA) <= len(setB)):
33             |       for key in setA:
34                 |           if(key in setB):
35                     |               setResult[key] = True
36             |   else:
37                 |       for key in setB:
38                     |           if(key in setA):
39                         |               setResult[key] = True
40             |   return setResult
41
42     def difference(setA, setB):
43         |   setResult = dict()
44         |   for key in setA:
45             |       if(not key in setB):
```

```

42     def difference(setA, setB):
43         setResult = dict()
44         for key in setA:
45             if(not key in setB):
46                 setResult[key] = True
47         return setResult
48
49     def complement(setA, universalSet):
50         return MySet.difference(universalSet, setA)
51
52     class Mapping:
53         operations = ["union", "intersection", "difference"]
54
55         def split(string):
56             splitted_string = string.split()
57             i = 0
58             while(i < len(splitted_string)):
59                 jump = 0
60                 while(splitted_string[i][0] == '(' and len(splitted_string[i]) != 1):
61                     splitted_string[i] = splitted_string[i][1:]
62                     splitted_string.insert(i, "(")
63                     i += 1
64                 while(splitted_string[i][-1] == ')' and len(splitted_string[i]) != 1):
65                     splitted_string[i] = splitted_string[i][: -1]
66                     splitted_string.insert(i+1, ")")
67                     jump += 1
68                 i += 1 + jump
69             return splitted_string
70
71         def mapSet(setsDict, key):
72             value = dict()
73             if(key in setsDict):
74                 value = MySet.createSet(setsDict[key])
75             else:
76                 raise Exception("No set called " + key)
77             return value
78
79         def mapOperation(operationName, setA, setB):
80             if(operationName == "union"):
81                 return MySet.union(setA, setB)
82             if(operationName == "intersection"):
83                 return MySet.intersection(setA, setB)
84             if(operationName == "difference"):
85                 return MySet.difference(setA, setB)
86             raise Exception("No operation named " + operationName)

```

```

79     def mapOperation(operationName, setA, setB):
80         if(operationName == "union"):
81             return MySet.union(setA, setB)
82         if(operationName == "intersection"):
83             return MySet.intersection(setA, setB)
84         if(operationName == "difference"):
85             return MySet.difference(setA, setB)
86         raise Exception("No operation named " + operationName)
87
88     def nodeEvaluation(instruction_flow, startIndex, setsDict):
89         currentNodeValue = dict()
90         currentOperation = "union"
91         index = startIndex
92
93         while(index < len(instruction_flow)):
94             if(instruction_flow[index] == '('):
95                 recievedNodeValue, index = MySet.Mapping.nodeEvaluation(instruction_flow, index+1, setsDict)
96                 currentNodeValue = MySet.Mapping.mapOperation(currentOperation, currentNodeValue, recievedNodeValue)
97             else:
98                 if(instruction_flow[index] == ')'):
99                     return currentNodeValue, index
100                 else:
101                     if(instruction_flow[index] in MySet.Mapping.operations):
102                         currentOperation = instruction_flow[index]
103                     else:
104                         recievedNodeValue = MySet.Mapping.mapSet(setsDict, instruction_flow[index])
105                         currentNodeValue = MySet.Mapping.mapOperation(currentOperation, currentNodeValue, recievedNodeValue)
106                 index += 1
107
108         return currentNodeValue, index
109
110     def evaluateExpression(expression, setsDict):
111         instruction_flow = MySet.Mapping.split(expression)
112         resultValue, index = MySet.Mapping.nodeEvaluation(instruction_flow, 0, setsDict)
113         return resultValue
114
115
116 # setsDict = {'A': [1,2,3], 'B': [3,4,5], 'C': [5,6,7], 'D': [3]}
117 # expression = "(A intersection (B union C)) difference D"
118 # 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 ")".