|  |
| --- |
| **Bekbolot kyzy Asel April 28, 2023**  **QECV0G**  [qecv0g@inf.elte.hu](mailto:qecv0g@inf.elte.hu)  Group 10 |
| **2nd Assignment/Task Number 3** |

**Task**

Layers of gases are given, with certain type (ozone, oxygen, carbon dioxide) and thickness, affected by atmospheric variables (thunderstorm, sunshine, other effects). When a part of one layer changes into another layer due to an athmospheric variable, the newly transformed layer ascends and engrosses the first identical type of layer of gases over it. In case there is no identical layer above, it creates a new layer on the top of the atmosphere. In the following we declare, how the different types of layers react to the different variables by changing their type and thickness. No layer can have a thickness less than 0.5 km, unless it ascends to the identical-type upper layer. In case there is no identical one, the layer perishes.

***A picture containing text, screenshot, font, line

Description automatically generated***

The program reads data from a text file. The first line of the file contains a single integer N indicating the number of layers. Each of the following N lines contains the attributes of a layer separated by spaces: type and thickness. The type is identified by a character: Z – ozone, X – oxygen, C – carbon dioxide. The last line of the file represents the atmospheric variables in the form of a sequence of characters: T – thunderstorm, S – sunshine, O – others. In case the simulation is over, it continues from the beginning.

The program should continue the simulation until one gas component totally perishes from the atmosphere. The program should print all attributes of the layers by simulation rounds!

The program should ask for a filename, then print the content of the input file. You can assume that the input file is correct. Sample input:

A picture containing text, screenshot, line, rectangle

Description automatically generated

***Set of values***

***Bag =*** {(a∈ ℤ, b∈

***Operations***

*1.Insert an element*

Inserting an integer to the bag. Bag is sequence of pairs, where the second one is the frequency of the element. Thus, if the element is already in the set, it should just increment the value of the frequency of an element. Otherwise, add the element to the list and set its frequency to 1.

Formally:

A = ( b: Bag, n: ℤ)

Pre = (b=b’)

Post =( b.elems = b.elems )

Remark:

*2. Remove an element*

When removing an element from the bag, the frequency of that element should also be decremented. Also, throw an error if the bag is empty or if the number is not in the bag.

Formally: A = ( b:Bag, n: ℤ)

Pre = (b=b’ ∧ n ∈ b ∧ |b|> 0)

Post =( b.elems= (b.elems \ {n}) ) )

*3. Return the frequency of the given element*

Frequent() function that will return the frequency of a given element (Getting the index of the given element from b.elems and returning the value from b.freqs at the given index). Throw an error if the given element is not in the bag and if the bag is empty.

Formally: A = ( b:Bag, n: ℤ, m: ℤ)

Pre = (b=b’ ∧ n ∈ b ∧ |b|> 0)

Post =(Pre ∧ m= b.frequent(n) ))

*4. Returning the largest element*

Throw an error if the bag is empty

Formally: A = ( b:Bag , max: ℤ)

Pre = (b=b’ ∧ |b|!=0)

|b|

Post =(Pre ∧ max= MAX b[i]) )

*i = 1*

*5. Printing the content of the bag*

Formally: A = (b:Bag)

Pre = (b=b’)

b.Count

Post =( PRINT b[i] )

*i = 1*

***Representation***

***Integers stored in the list1 and its frequencies stored in another list2.***

***Ex.: List <> elems = {1,2,3,4,6}***

***List <> freqs = {1,2,1,1,1}***

***For instance, the frequency of 2 is 2.***

***Implementation***

1 .Inserting an element

*Text, letter

Description automatically generated*

2.Removing an element

A picture containing diagram

Description automatically generated

3.Returning the frequency of a given element

Letter

Description automatically generated

4.Returning the largest element of the bag

Diagram

Description automatically generated

*5.* Printing the bag

*A piece of paper with writing on it

Description automatically generated*

**Auxiliary operations:**

6. Getting and index of the given element

Text, letter

Description automatically generated

7. IsIn(n)

Diagram

Description automatically generated with low confidence

*Testing*

*Testing the operations*

*1.TestInsert()*

*This method tests the InsertElement() function of the Bag class, which inserts an element into the bag.*

*It creates 4 Bag objects and performs various insertions to test the functionality of the method.*

1. *Put 2,2,1 and check the size of the bag and frequency of 2*
2. *Create empty bag and check the size of it*
3. *Create a bag and put elements using constructor*
4. *Create a bag and put 0s using constructor and check if it works*

*2.TestRemove()*

1. *Remove unique element from the bag*
2. *Remove duplicated element*
3. *Test the case if the bag is empty*
4. *Test the case, when user inputs an element which is not in the bag*

*3.TestReturnFreq()*

1. *Check the frequency of a duplicated element*
2. *Check the case when user inputs an element which is not in the bag*
3. *Check the case when bag is empty*

*4.TestMaxElem( )*

1. *Check the maximum element, which appears twice*
2. *Remove the previous max element and check again*
3. *Check the case when the bag is empty*

*5.TestGetIndex()*

*Testing auxiliary method getIndex()*

*6.TestContains()*

*Testing auxiliary method Contains()*