16th March 2019

**1. assignment/10. task**

**Sadi Mamedov**

LHXOTB

[sadimamedov7@gmail.com](mailto:treszka@g.com)

Group 2

# Task

*Implement the bag type which contains integers. Represent the bag as a sequence of (element, frequency) pairs. Implement as methods: inserting an element, removing an element, returning the frequency of an element, returning the largest element in the bag (suggestion: store the largest element and update it when the bag changes), printing the bag.*

# Bag type of integers

## Set of values

*Item* (*elem , freq*) = {*elem : e*  *freq : f* | *e*  ℤ  *f*  ℕ+}

## Representation

Pairs of items has to be strored in dynamic container vec : vector <int>.

*Item* (*e , f*) : Bag vec = < (*e1 , f1) ,( e2 , f2) ,( e3 , f3) …>*

## Implementation1

## 1. Insering an element

## Bag is represented by Item (elem , freq) pairs where a stands for inserted element (a : Item) ,

## n stands for the size of Bag. Operator remove indicates erasing corresponding element of type vec.

## In case one element added more than once, it resulted an increment in frequency of that element.

## 

## 2. Removing an element

## Removing an element removes the pair whose element part of type integer equals the user input rm of type integer and a stands for returned Item type (a : Item) ,n stands for the size of bag. Operator remove indicates erasing corresponding element of type vec. In case inserted input is different than any element of bag, program terminates with type error.

## 

## 3. Frequency of an element

## Frequency of element gives back frequency of inserted user input which is represented by el of type integer ,n stands for the size of bag and r type integer stands for freq unit of pair Item (elem , freq).

## 

## 4. Largest element

## In case bag is not empty, this program returns the pair having largest element, otherwise program terminates with error. Integer type maxi indicates starting index, while n stands for the size of bag.

## 

## 5. Printing the Bag

## Printing the bag prints Item(elem, freq) type members of the bag. Output stream is represented by operator cout , where ‘< >’ stands for empty buffer , and n stands for size of the bag.

## 

## Class

The Bag type is worked out as a class. The size of the bag is undefined. At the operations, elements should have same data type, otherwise the program should throw an exception.

|  |  |  |
| --- | --- | --- |
| **Bag** |  | **Exceptions** |
| - vec : Item[(elem,freq)….] |  |
| - elem : int  - freq : +int |
| + Bag() | + EMPTY\_BAG |
| + get(Item) : Bag | + INVALID\_ELEMENT |
| + frequency (int) : int |  |
| + largest () : Item |  |
| + remove (int) : Item |  | |
| + insert(Item) : void |
| + ostream() : Bag |

The entries of the Item can be represented as a vector<int>. The constructor is default and allocates the dynamic vector<int>.

Operator ostream is created as a friend function and the class is extended by getter.

For error handling, two exceptions are defined. EMPTY\_BAG is thrown in case of operation (largest) when there exists no element in the bag. INVALID\_ELEMENT is raised in case of operation (remove) when an inserted element does not match the bag member element.

# Testing

Testing the operations (black box testing)

1. Empty status.
   1. Creating bag object, checking empty status after operations insert, remove.
2. Inserting operator
   1. Inserting a single item, checking the length and getting an item.
   2. Creating multiple items, inserting items, checking the size and getting all of items through getter.
   3. Inserting the same member item, checking consistency of size, and frequency.
3. Maximum value of bag
   1. Checking empty status of bag, in empty case testing throw of exception.
   2. Creating one item, getting the maximum value, as well as testing size of bag after removing, inserting an item.
   3. Inserting multiple items, getting the maximum values varying on their positions in bag (last, middle, first) and testing the precise addresses through getter.
4. Frequency of an element
   1. Creating three items, inserting two distinct items and checking each one’s frequency through freq unit of type Item.
   2. Inserting same item into bag, checking increment on similar item member’s frequency, and the size after removing an item.
5. Removing operator
   1. Checking an inserted irrelevant item to members of bag in empty status and having single item., where in both cases testing throw of exception.
   2. Inserting multiple items, checking removed and remaining items using getter, afterwards checking the size of bag.

Testing based on the code (white box testing)

1. Creating an extreme-size vector (100000).
2. Generating and catching all exceptions.