JOHANNES KEPLER
UNIVERSITÄT LINZ

Algorithms and Data Structures 2
Winter term 2021
Martin Schobesberger, Stefan Grünberger
Dari Trendafilov, Markus Weninger

Assignment 03

Deadline: **Tue. 07.12.2021, 23:59**

Hashing

Please, don't change any skeleton we provide. Of course, you are allowed to add code (methods and global variables), as long as you don't break the skeleton (method names and variables' types) we need for our tests to work correctly. Please remember to fill out the time log survey in Moodle.

1. Chaining 12 points

Implement the class `ChainingHashSet`, which internally uses a hash table for storing data. The capacity of a new hash set (i.e. the size of the list to which the hash values are mapped) has to be defined in the constructor. The hash table has to work according to the *chaining* principle and therefore store colliding elements in an overflow list.

The hash function *h* is defined as

h(k) = k mod N

where *k* is the key and *N* is the length of the hash table. The modulo division can be achieved by the modulo operation '**%**'.

```
class ChainingHashSet():

    def get_hash_code(self, key):
        """Hash function that calculates a hash code for a given key using the modulo division.
        :param key:
              Key for which a hash code shall be calculated according to the length of the hash table.
        :return
              The calculated hash code for the given key.
        """
        pass

    def get_hash_table(self):
        """(Required for testing only)
        :return the hash table.
        """
        pass

    def set_hash_table(self, table):
        """(Required for testing only) Set a given hash table
        :param table:
              Given hash table which shall be used.
        """
        pass

    def get_table_size(self):
        """returns the number of stored keys (keys must be unique!).
        """
        pass

    def insert(self, key):
        """Inserts a key and returns True if it was successful. If there is already an entry with the
          same key, the new key will not be inserted and False is returned.
         :param key:
              The key which shall be stored in the hash table.
         :return:
              True if key could be inserted, or False if the key is already in the hash table.
         :raises:
              a ValueError if any of the input parameters is None.
         """
        pass
```

```
def contains(self, key):
        """Searches for a given key in the hash table.
         :param key:
               The key to be searched in the hash table.
         :return:
               True if the key is already stored, otherwise False.
         :raises:
               a ValueError if the key is None.
         """
        pass

    def remove(self, key):
        """Removes the key from the hash table and returns True on success, False otherwise.
        :param key:
              The key to be removed from the hash table.
        :return:
              True if the key was found and removed, False otherwise.
        :raises:
           a ValueError if the key is None.
        """
        pass

    def clear(self):
        """Removes all stored elements from the hash table by setting all nodes to None.
        """
        pass

    def to_string(self):
        """Returns a string representation of the hash table (array indices and stored keys) in the format
          Idx_0 {Node, Node, ... }, Idx_1 {...}
         e.g.: 0 {13}, 1 {82, 92, 12}, 2 {2, 32}, """
        pass
```

For implementing the overflow list use nodes of the class `ChainingHashNode`*.* These nodes store a key and a reference to the next node in the chain.

```
class ChainingHashNode():
   def __init__(self, key = None):
      self.key = key
      self.next = None
```

JOHANNES KEPLER UNIVERSITÄT LINZ

Algorithms and Data Structures 2
Winter term 2021
Martin Schobesberger, Stefan Grünberger
Dari Trendafilov, Markus Weninger

Assignment 03

Deadline: **Tue. 07.12.2021, 23:59**

2. Double Hashing — 12 points

Apply the **double hashing** algorithm presented in the exercise using **pen & paper**.

In contrast to example 1, this algorithm doesn't need additional memory for the insertion of colliding elements. In case of collisions no overflow list is used, instead the elements are stored at other vacant positions in the hash table using a 2nd hash function. The two hash functions *h1* and *h2* are defined as

h1(k) = k mod N
h2(k) = 1+ k mod (N-1)

where *k* is the key and *N* is the length of the hash table.

Resolve any collisions by (repeatedly) applying the following probing sequence:

h1 = (h1 + h2) % N

a) Prefilling of table

Fill the following table using the provided insert statements:

For each digit also provide the probing sequence (indices separated by commas).

insert(19)

0	1	2	3	4	5	6	7	8	9	10	11	12

Probing sequence:

insert(6)

0	1	2	3	4	5	6	7	8	9	10	11	12

Probing sequence:

JOHANNES KEPLER
UNIVERSITÄT LINZ

Algorithms and Data Structures 2
Winter term 2021
Martin Schobesberger, Stefan Grünberger
Dari Trendafilov, Markus Weninger

Deadline: **Tue. 07.12.2021, 23:59**

Assignment 03

insert(11)

0	1	2	3	4	5	6	7	8	9	10	11	12

Probing sequence:

insert(26)

0	1	2	3	4	5	6	7	8	9	10	11	12

Probing sequence:

insert(14)

0	1	2	3	4	5	6	7	8	9	10	11	12

Probing sequence:

Assignment 03

b) Insert of student ID

Fill the table, that is now prefilled with keys from part a), with the three rightmost digits of your student id. Again, note the probing sequence.
Please make a note in case one of the digits of your student id can not be entered.

e.g.: student id = k12345**678** -> *insert(8), insert(7), insert(6)*

Student id: k X X X X X ___ ___ ___
d6 d7 d8

insert(d8)

0	1	2	3	4	5	6	7	8	9	10	11	12

Probing sequence:

insert(d7)

0	1	2	3	4	5	6	7	8	9	10	11	12

Probing sequence:

Algorithms and Data Structures 2
Winter term 2021
Martin Schobesberger, Stefan Grünberger
Dari Trendafilov, Markus Weninger

Deadline: **Tue. 07.12.2021, 23:59**

Assignment 03

insert(d6)

0	1	2	3	4	5	6	7	8	9	10	11	12

Probing sequence:

Submission

Put your source files and a pdf for part 2 into a ZIP archive and name the file **k12345678-assignment03.zip** where k12345678 should reflect your student ID.