Data Structures and Algorithms
Spring 2021
Assignment №1

IT University Innopolis

Contents

1 About this homework **2**
- 1.1 Coding part . . . 2
  - 1.1.1 Preliminary tests . . . 2
  - 1.1.2 Code style and documentation . . . 2
- 1.2 Theoretical part . . . 2
- 1.3 Submission . . . 3

2 Coding part **4**
- 2.1 Sorted List ADT . . . 4
- 2.2 Managing Pawn Shop Items . . . 5
- 2.3 Accounting for a café . . . 6

3 Theoretical part **8**
- 3.1 Big-O notation . . . 8
- 3.2 Asymptotic complexity . . . 8
- 3.3 Recurrences and Master Theorem . . . 8

1 About this homework

1.1 Coding part

For practical (coding) part you have to provide you program as a single Java class file that contains all necessary definitions and a `main` method.

The program should read from standard input and write to standard output unless mentioned otherwise.

1.1.1 Preliminary tests

For some coding parts a CodeForces system will be used for preliminary tests. You will have to specify a submission number in those systems as a proof that your solution is correct. Only correct solutions will be assessed further by TAs.

1.1.2 Code style and documentation

The source code should contain adequate internal documentation in the form of comments. Internal documentation should explain how the code has been tested, e.g. the various scenarios addressed in the test cases.

Do not forget to include your name in the code documentation!

All important exceptions should be handled properly.

Bonus code style points might be awarded for elegant solutions, great documentation and the coverage of test cases. However, these bonus code style points will only be able to cancel the effects of some penalties.

1.2 Theoretical part

Solutions to theoretical exercises have to be elaborate and clear. Write all solutions in a single document, then save or compile it as a PDF for submitting.

Do not forget to include your name in the document!

1.3 Submission

You are required to submit your solutions to all parts via Moodle.

For this assignment you will need to submit:

- Java class file(s) for the coding parts;
- link(s) to accepted submission(s) in CodeForces as a proof of correctness;
- a PDF file with solutions for theoretical parts.

Submit files as a single archive with your name and group number. For example, if your name is *Ivanov Ivan* and your group is *BS20-05* then your archive should be named `BS20-05_Ivanov_Ivan.zip`.

2 Coding part

2.1 Sorted List ADT

Warning: In this assignment you are not allowed to use data structures from standard libraries (such as `java.util`*).*

Consider a "sorted list" ADT with the following interface:

```
interface SortedList<T extends Comparable<T>> {
   void add(T item);     // add a new item to the List
   T least();            // return the least element
   T greatest();         // return the greatest element
   T get(int i);         // return the i-th least element
   int indexOf(T item); // return the index of an element (in a sorted sequence)
   void remove(int i);  // remove i-th least element from the list
   List<T> searchRange(T from, T to);  // find all items between from and to
   int size();           // return the size of the list
   boolean isEmpty();    // return whether the list is empty
 }
```

Note: you can declare List ADT in Java and C++ as either an interface or an abstract class.

You need to

1. Declare a `SortedList<T>` interface.
2. Provide an implementation of `SortedList<T>` with one of the following:
3. `LinkedSortedList<T>` — an implementation based on singly linked list;
4. `DoublyLinkedSortedList<T>` — an implementation based on doubly linked list;
5. `ArraySortedList<T>` — an implementation based on arrays.
6. With any option you choose make sure that time complexity for `least()` and `greatest()` is $O(1)$ worst case.
7. Write down the time complexity for all implemented interface methods, specifying if it is given for *worst case* or *amortized*. Provide a short justification for the given time complexity for `searchRange(from, to)` method.

2.2 Managing Pawn Shop Items

Warning: In this assignment you are not allowed to use data structures from standard libraries (such as `java.util`*).*

Using your implementation of `SortedList<T>` write a program that manages a list of items sold at a pawn shop. A pawn shop frequently updates the list of items it stores and can provide a list of sold items in any price range.

The first line of input contains a single number N ($0 < N < 10000$). Each of the following N lines contains one of the following commands:

- `ADD <price> <item>` — add an item with a given price;
- `REMOVE <price> <item>` — remove an item;
- `LIST <min price> <max price>` — ask for a list of items in the given price range.

Sample input:

```
9
ADD $150.00 Watch
ADD $10.00 Old T-shirt
ADD $300.00 TV-set
LIST $50.00 $500.00
REMOVE $150.00 Watch
LIST $50.00 $500.00
ADD $130.00 A very nice coffee mug
ADD $384.00 A small UFO
LIST $50.00 $350.00
```

The output should contain a single line for every `LIST` command in the input. The line should contain the list of items with prices, separated by comma: `<price1> <item1>, <price2> <item2>, ..., <priceN> <itemN>`. The list should be in order of increasing price. Sample output:

```
$150.00 Watch, $300.00 TV-set
$300.00 TV-set
$130.0 A very nice coffee mug, $300.00 TV-set
```

2.3 Accounting for a café

Warning: In this assignment you are not allowed to use data structures from standard libraries (such as `java.util`).

A local café would like to keep track of its earnings. Your task is to implement a simple accounting program for the café. The program is given a sequence of entries, one per sold item. Each entry contains a timestamp, receipt number, sold item title and its cost. The task is to compute for each date (year, month and day) total amount of money earned, as well as number of unique customers (number of unique receipts).

First line of the input contains a single number N — the number of entries in the input. Following N lines each describe one entry. Each entry consists of 5 fields separated by spaces:

- date (`YYYY-MM-DD` format),
- time (`HH:MM:SS` format),
- receipt ID (`#<ID>` format, where ID can contain digits, letters and dash (`-`)),
- cost (`$<number>` format) and
- item title (no specified format).

Entries are not guaranteed to appear in chronological order. Sample input:

```
7
2021-02-03 01:02:03 #123-AC $250.00 Latte Large
2021-02-03 03:02:03 #133-AC $150.00 Latte Small
2021-02-03 01:02:03 #123-AC $550.00 Espresso
2021-02-03 01:02:03 #123-AC $253.00 Flat White
2021-02-05 01:02:03 #143-AC $54.00 Double Espresso
2021-02-05 01:02:03 #143-AC $5.00 Sugar
2021-02-03 01:02:03 #123-AC $123.00 Brownie
```

The program should output K lines, where K is the number of different dates in the input. For each date the program should output a line with date (`YYYY-MM-DD`), total cost (`$<number>`) and number of unique receipt IDs (`<number>`). Output is not required to appear in chronological order. Sample output:

```
2021-02-03 $1326.00 2
```

```
2021-02-05 $59.00 1
```

Implement this program using a custom implementation of a HashMap:

1. Specify `Map` ADT using a generic interface or a generic abstract class. Map ADT should provide all the important operations that you will need to implement a solution to the accounting problem.
2. Implement the `HashTable` class (implementing the `Map` ADT) that uses the hash-function, and a combination of Separate Chaining with Linear Probing as the collision handling strategy, and implements all important operations.
3. The main program should read the standard input which contains a list of entries and use implemented `HashTable` to store them. You should associate a collection of entries to each date.
4. After constructing such a `HashTable`, the program should, for each date, output the required information (total cost and number of unique receipt IDs).
5. All exceptions, errors, and issues must be caught and handled.

3 Theoretical part

3.1 Big-O notation

Use big-O notation to summarize the asymptotic behavior of the following functions. For full grade, you must provide the details about your calculation.

1. $n^2 + 10n \log n + 50n + 100$
2. $n^{\frac{7}{2}} + 7n^3 \log n + n^2$
3. $6^{n+1} + 6(n+1)! + 24n^{42}$

3.2 Asymptotic complexity

Write down the asymptotic complexity of the following functions in terms of big-O notation. For full grade, you must provide the details about your calculation.

```
void calculateComplexity(int n) {
  for (int i=n; i>0; i--)
    for (int j=0; j<i; j++)
      checkpoint(n);

  void checkpoint(int number) {
    for (int i=1; i<number; i=i*3)
      System.out.println("DSA Homework "+i);
  }
}
```

3.3 Recurrences and Master Theorem

Running time $T(n)$ of processing n data items with some unspecified algorithm is described by the following recurrence:

$$T(n) = \sqrt{k} \cdot T\left(\frac{n}{k^2}\right) + c \cdot \sqrt[4]{n}; \quad T(1) = 0$$

What is the computational complexity of this algorithm (express complexity in a closed form, i.e. without recurrence, in terms of c, n and k)? Which case of Master Theorem applies and why? The answer should necessarily include calculations to justify your claim.