Gebze Technical University
Department of Computer Engineering CSE 222/505 - Spring 2020
Homework 7 Report

Alper YAŞAR
151044072

1 – Q3

1. SYSTEM REQUIREMENTS

Modified the skip list implementation in the book so that each node in the lowest-level list keeps several elements instead of just one element as in a B-tree node.

1. For each node `root`, the keys are stored in increasing order.
2. In each node, there is a boolean value `root.leaf` which is true if root is a leaf.
3. If `order` is the order of the tree, each internal node can contain at most `order - 1` keys along with a pointer to each child.
4. Each node except root can have at most order children and at least `order/2` children.
5. All leaves have the same depth
6. The root has at least 2 children and contains a minimum of 1 key.

Searching

Searching for an element in a B-tree is the generalized form of searching an element in a Binary Search Tree. The following steps are followed.

1. Starting from the root node, compare key with the first key of the node.
   If `key = the first key of the node`, return the node and the index.
2. If `key.leaf = true`, return `NULL`
3. If `key < the first key of the root node`, search the left child of this key recursively.
4. If there is more than one key in the current node and `key > the first key`, compare k with the next key in the node.
   If `key < next key`, search the left child of this key
   Else, search the right child of the key.
5. Repeat steps 1 to 4 until the leaf is reached.

Insertion Operation

1. If the tree is empty, allocate a root node and insert the key.
2. Update the allowed number of keys in the node.
3. Search the appropriate node for insertion.
4. If the node is full, follow the steps below.

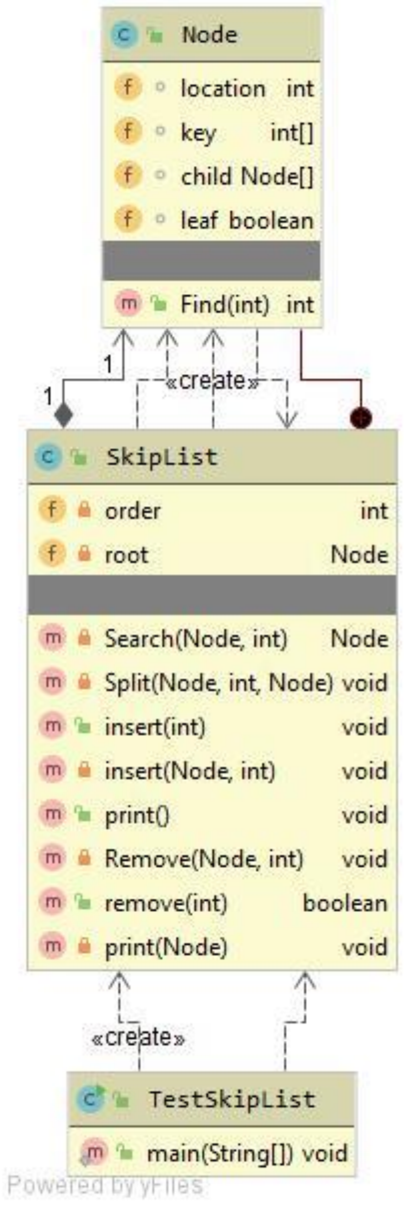
5. Insert the elements in increasing order.
6. Now, there are elements greater than its limit. So, split at the median.
7. Push the median key upwards and make the left keys as a left child and the right keys as a right child.
8. If the node is not full, follow the steps below.
9. Insert the node in increasing order.

Deletion Operation

Before going through the steps below, one must know these facts about a B tree of degree **order.**

1. A node can have a maximum of order children. (i.e. 3)
2. A node can contain a maximum of `order - 1` keys. (i.e. 2)
3. A node should have a minimum of `⌈order 2⌉` children. (i.e. 2)
4. A node (except root node) should contain a minimum of `⌈order 2⌉ - 1` keys.

2. Class Diagram



3. PROBLEM SOLUTION APPROACH

The maximum and the minimum number of elements in a node should be specified during the construction of the data structure. The number of elements at each node should be smaller than the maximum. The number of elements at each node should be larger than the minimum, except in the case of one node skip list.

Insert the new element into the corresponding node. Split the node when it is necessary.

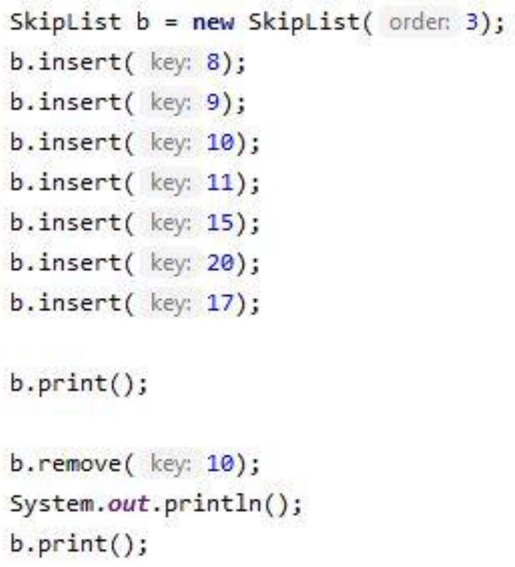
After a deletion operation, the number of elements in a node may become smaller than the minimum. Combine the node with its predecessor or successor.

4. TEST CASES

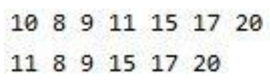
Test Case	Test Scenerio	Test Data	Expected Result
1. inser	Insert as parameter	8 10	true
2. remove	Remove a key from tree	10	true
3. search	Search a key is exist in tree	8	8 Else null
4. print	Print all nodes to terminal		print

Tested with given examples in pdf.

5. RUNNING AND RESULTS



```
SkipList b = new SkipList( order: 3);
b.insert( key: 8);
b.insert( key: 9);
b.insert( key: 10);
b.insert( key: 11);
b.insert( key: 15);
b.insert( key: 20);
b.insert( key: 17);

b.print();

b.remove( key: 10);
System.out.println();
b.print();
```



```
10 8 9 11 15 17 20
11 8 9 15 17 20
```

3 – Q3

Skip-List in Q-2

İnsert 10000: 88, 59, 54, 67, 77, 106, 91, 80, 64, 62

İnsert 20000: 388, 420, 207, 338, 327, 350, 273, 255, 251, 257

İnsert 40000: 1895, 1037, 1357, 1195, 1209, 1202, 1162, 1212, 1215, 1199

İnsert 80000: 4464, 4132, 5306, 3407, 5576, 3620, 6066, 6026, 4318, 5910

Remove 10000: 1, 1, 1, 1, 0, 0, 0, 0, 1, 2

Remove 20000: 5, 1, 1, 1, 1, 1, 0, 0, 0, 0

Remove 40000: 7, 1, 1, 1, 1, 0, 1, 1, 1, 0

Remove 80000: 3, 1, 3, 0, 2, 3, 1, 2, 1, 1

Skip-List in Book

İnsert 10000: 16, 8, 12, 9, 10, 9, 9, 7, 7, 11

İnsert 20000: 29, 21, 25, 22, 19, 16, 16, 14, 10, 13

İnsert 40000: 88, 67, 62, 176, 73, 46, 63, 107, 84, 73

İnsert 80000: 159, 142, 115, 180, 111, 113, 92, 112, 120, 108

Remove 10000: 2, 0, 0, 2, 0, 0, 1, 0, 0, 0

Remove 20000: 6, 0, 2, 1, 2, 1, 0, 1, 1, 0

Remove 40000: 5, 1, 2, 2, 1, 1, 0, 0, 1, 1

Remove 80000: 3, 2, 2, 1, 2, 0, 1, 2, 1, 1

Skip-List in Java

İnsert 10000: 32, 20, 9, 40, 6, 4, 7, 15, 12, 10

İnsert 20000: 57, 57, 19, 23, 17, 25, 12, 17, 19, 17

İnsert 40000: 83, 53, 40, 35, 40, 50, 34, 33, 61, 66

İnsert 80000: 196, 114, 145, 107, 92, 115, 73, 65, 73, 87

Remove 10000: 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

Remove 20000: 1, 0, 0, 0, 0, 0, 1, 0, 0, 0

Remove 40000: 1, 1, 1, 0, 0, 0, 0, 0, 0, 0

Remove 80000: 0, 1, 1, 1, 1, 0, 0, 0, 0, 0

Red Black tree implementation in java

İnsert 10000: 13, 14, 7, 11, 8, 11, 10, 7, 6, 7

İnsert 20000: 25, 39, 25, 20, 37, 19, 18, 10, 9, 8

İnsert 40000: 38, 39, 31, 33, 40, 29, 19, 17, 10, 14

İnsert 80000: 101, 85, 89, 49, 108, 42, 42, 48, 50, 39

Remove 10000: 0, 1, 0, 0, 0, 0, 0, 0, 0, 0

Remove 20000: 0, 0, 0, 0, 1, 0, 0, 0, 0, 0

Remove 40000: 1, 1, 0, 0, 0, 0, 1, 0, 0, 0

Remove 80000: 1, 0, 0, 0, 0, 0, 0, 0, 0, 0

Red Black tree implementation in the Book

İnsert 10000: 16, 7, 5, 5, 5, 3, 6, 4, 3, 3

İnsert 20000: 35, 11, 8, 14, 22, 13, 9, 9, 7, 6

İnsert 40000: 66, 30, 29, 28, 23, 24, 22, 19, 16, 21

İnsert 80000: 109, 70, 57, 64, 72, 55, 50, 55, 47, 42

Remove 10000: 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

Remove 20000: 0, 0, 0, 0, 1, 0, 0, 0, 0, 0

Remove 40000: 0, 1, 0, 0, 0, 0, 0, 0, 0, 0

Remove 80000: 0, 0, 0, 0, 0, 0, 0, 0, 1, 0

Binary Search Tree implementation in the Book

İnsert 10000: 13, 5, 4, 4, 3, 4, 4, 3, 2, 3

İnsert 20000: 22, 7, 7, 7, 9, 10, 6, 8, 7, 6

İnsert 40000: 34, 19, 19, 15, 22, 14, 16, 20, 14, 11

İnsert 80000: 66, 58, 40, 51, 42, 39, 47, 63, 44, 53

Remove 10000: 0, 0, 1, 0, 0, 0, 0, 0, 0, 0

Remove 20000: 1, 0, 0, 0, 0, 0, 0, 0, 1, 1

Remove 40000: 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

Remove 80000: 0, 0, 0, 0, 1, 0, 0, 0, 0, 0

B-Tree implementation in the Book

İnsert 10000: 17, 8 ,12, 9, 6, 12, 7, 9, 11, 10

İnsert 20000: 27, 17, 12, 14, 16, 10, 9, 13, 17, 16

İnsert 40000: 46, 29, 31, 56, 32, 39, 41, 26, 34, 31

İnsert 80000: 96, 78, 60, 71, 62, 99, 102, 73, 64, 83

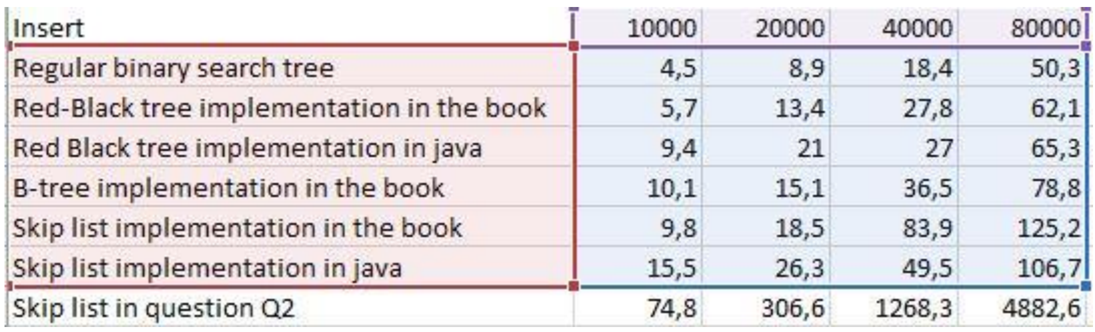
Remove 10000: 0, 0, 1, 0, 0, 0, 0, 0, 0, 0

Remove 20000: 1, 0, 0, 0, 0, 0, 0, 0, 0, 0

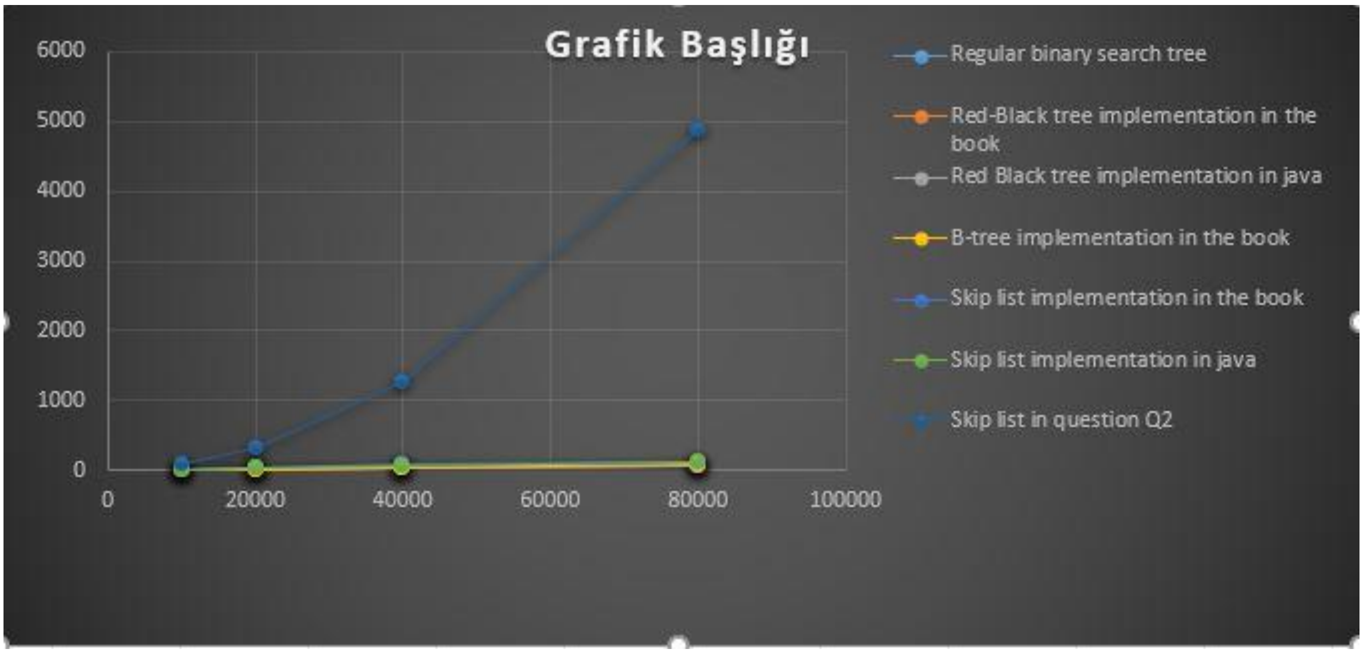
Remove 40000: 0, 0, 0, 0, 0, 0, 1, 0, 1, 0

Remove 80000: 2, 0, 1, 0, 1, 0, 1, 0, 0, 0

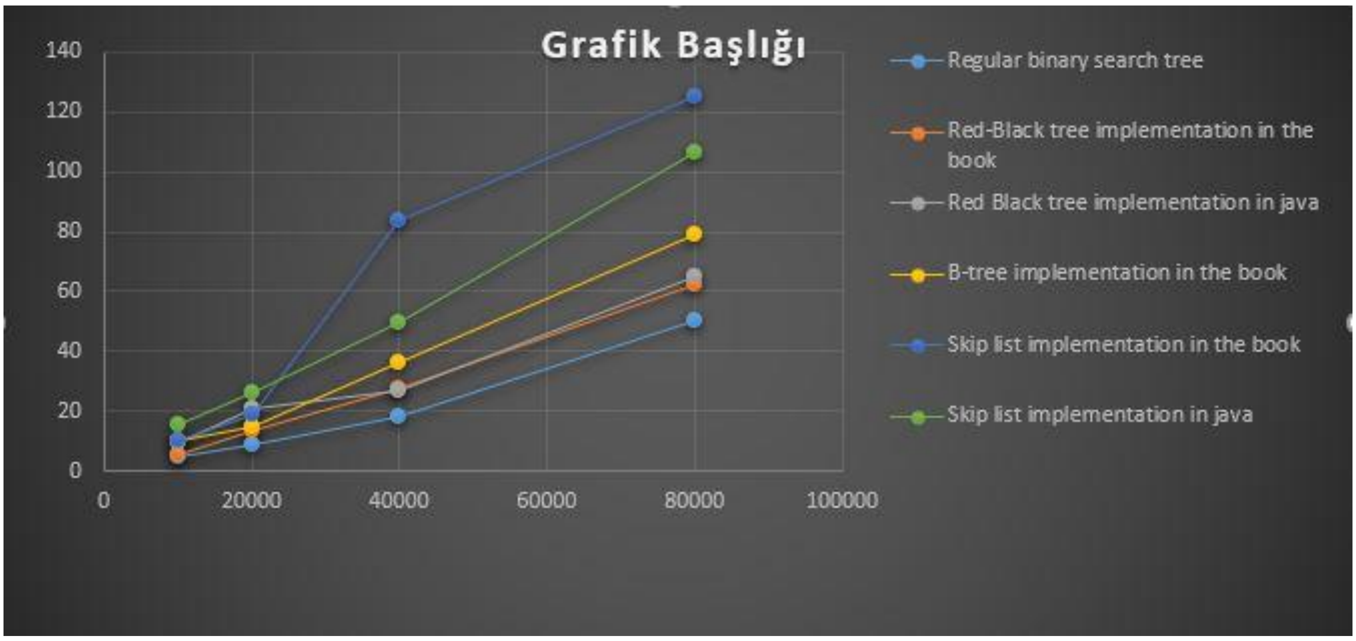
Insert Table



Insert	10000	20000	40000	80000
Regular binary search tree	4,5	8,9	18,4	50,3
Red-Black tree implementation in the book	5,7	13,4	27,8	62,1
Red Black tree implementation in java	9,4	21	27	65,3
B-tree implementation in the book	10,1	15,1	36,5	78,8
Skip list implementation in the book	9,8	18,5	83,9	125,2
Skip list implementation in java	15,5	26,3	49,5	106,7
Skip list in question Q2	74,8	306,6	1268,3	4882,6

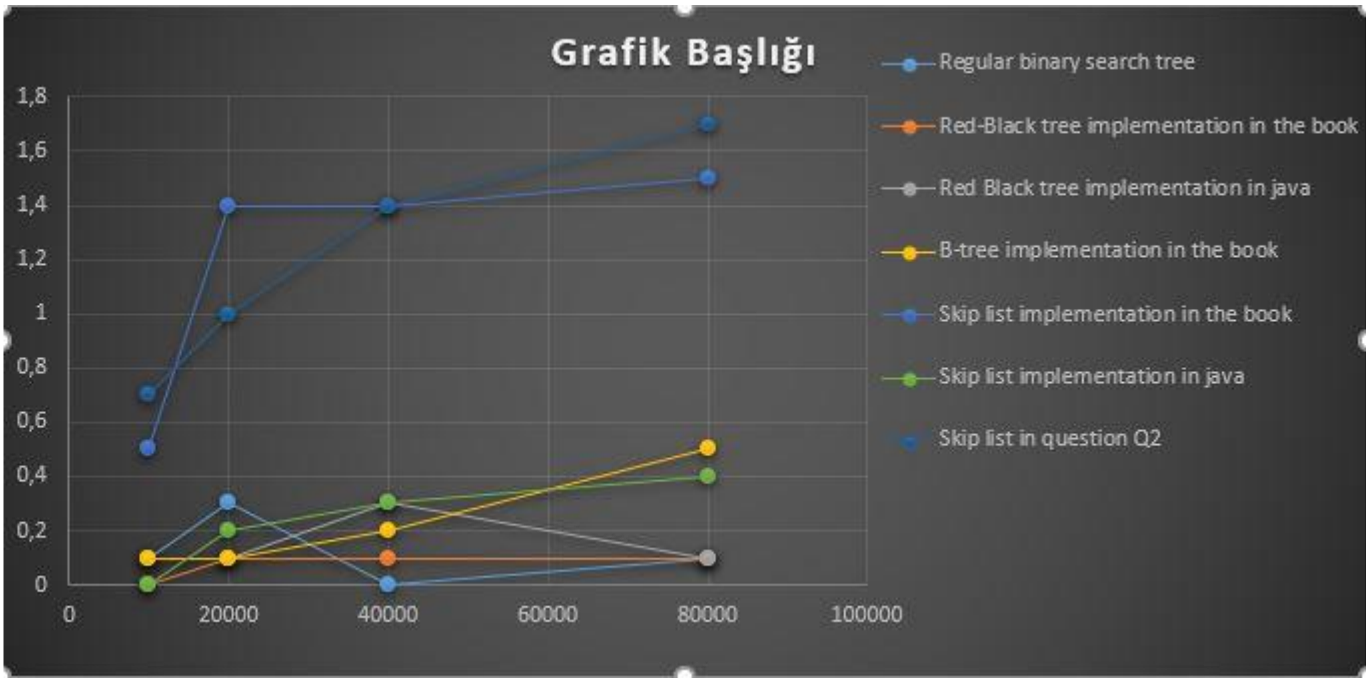


With Skiplist



Without skiplist seeing the other distribution on table

Remove	10000	20000	40000	80000
Regular binary search tree	0,1	0,3	0	0,1
Red-Black tree implementation in the book	0	0,1	0,1	0,1
Red Black tree implementation in java	0,1	0,1	0,3	0,1
B-tree implementation in the book	0,1	0,1	0,2	0,5
Skip list implementation in the book	0,5	1,4	1,4	1,5
Skip list implementation in java	0	0,2	0,3	0,4
Skip list in question Q2	0,7	1	1,4	1,7



1 – Q4

1. SYSTEM REQUIREMENTS

Implemented a menu-driven program for managing a software store. The system has two types of users: administrators who can update information and users who browse software.

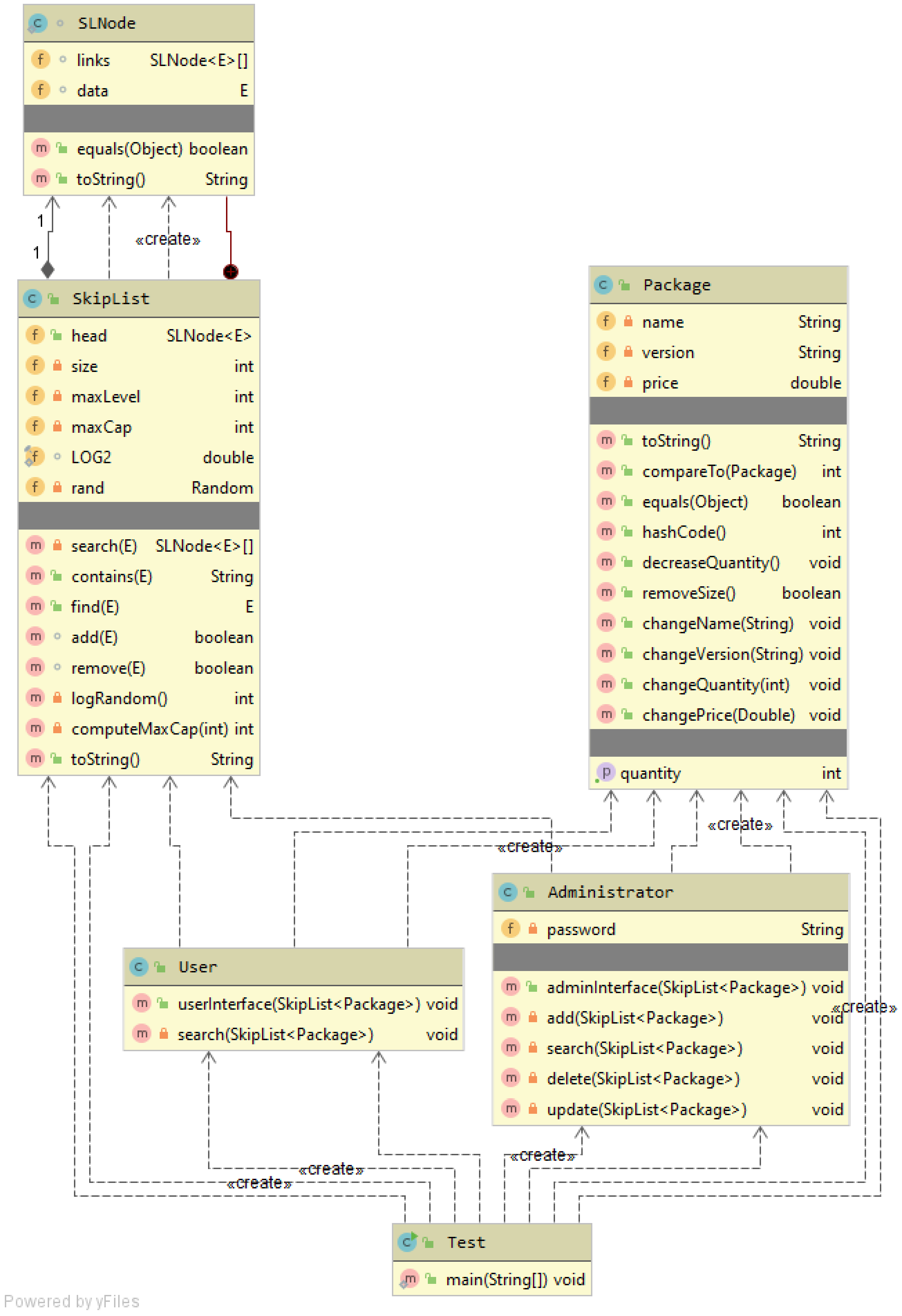
To update the information, administrators have to enter the system with a password (a single general password is sufficient). Software search is a public feature.

The system maintains the information about each software which at least includes name (including version), quantity, and price.

Administrator enters the system with a password, to be able to add, delete, update information. - Search by name, quantity etc. should be available.

Program automatically create a data structure including the following software packages; Adobe Photoshop 6.0, Adobe Photoshop 6.2, Norton 4.5, Norton 5.5, Adobe Flash 3.3, Adobe Flash 4.0.

2. Class Diagram



3. PROBLEM SOLUTION APPROACH

1. adminInterface()

When the enter administrator firstly ask password. If it is true then continue to system. Listing the administrator what can do in this system and ask to choose one. Search packages; search by name and version and, search by name all contains.
Delete package; decrease quantity if bigger than 1. Other case delete package information if quantity equal to 1 so when decrease it will be 0.
change information; firstly asking which package information will change. When matched with int the system asking name, version, quantity and price information wanted really changes. If admin said yes enter new info.
add new package with all information.

2. userInterface()

He/she can only search a package.

3. Package class

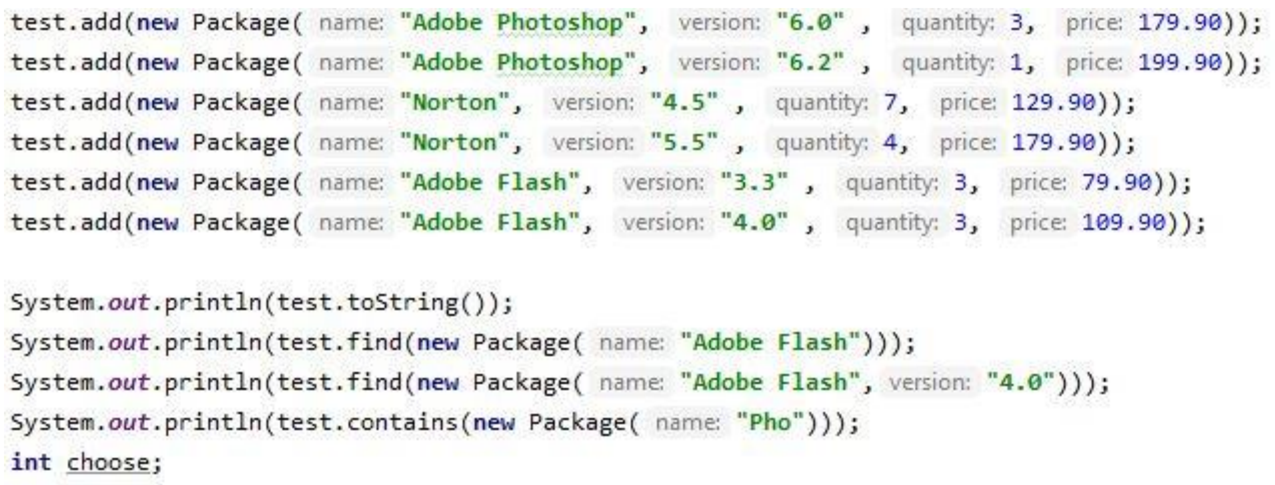
Hold the information about package; name, version, quantity, price.

4. TEST CASES

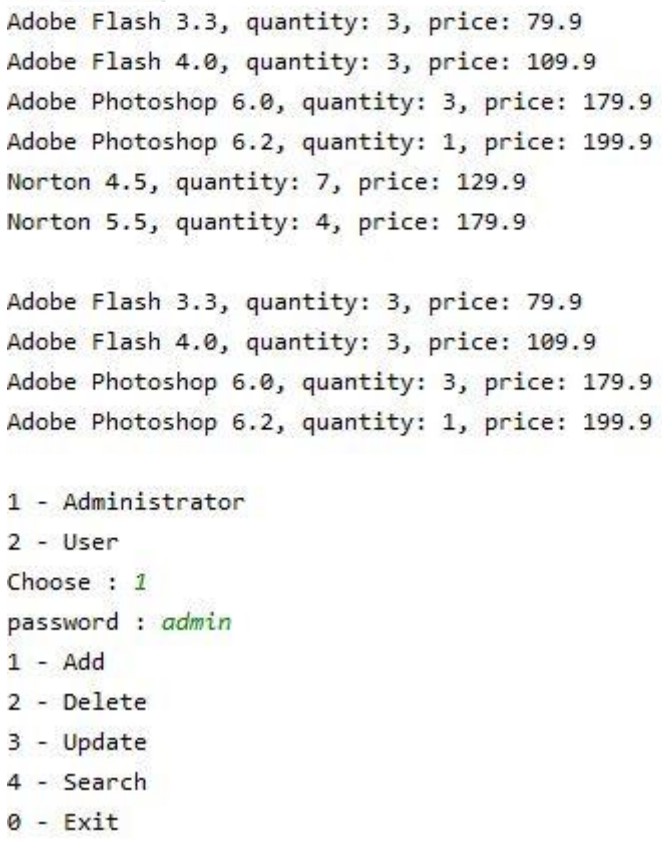
Test Case	Test Scenerio	Test Data	Expected Result
1. userInterface	When user choose “1” print user menu	1	User menu
2. adminInterface	When user choose “2” print admin menu	2	Administrator menu
3. search	Search in Skiplist by name or with version	Adobe	Listing all package includes “Adobe”
4. search	Search in Skiplist by name or with version	Adobe Photoshop, 6.2	Just showing info about Adobe Photoshop, 6.2

5. add	Add new package to the system. Write about its info: “name of package”, “version”, “quantity” and “price”	“Google Chrome” “4.0” “7” “121,21”	Create package and add to tree
6. remove	Taking exact name and matched version	“Adobe Flash” “4.0”	Decrease if quantity bigger than 1 or delete it.
8. update	Ask to admin what change want: name, version, price, quantity		Firstly remove package from tree and re-add to tree.

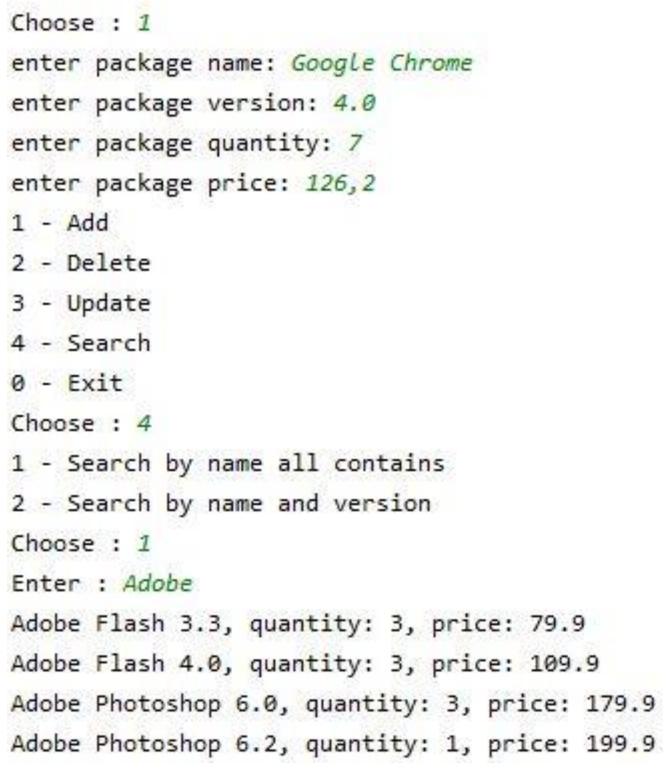
5. RUNNING AND RESULTS



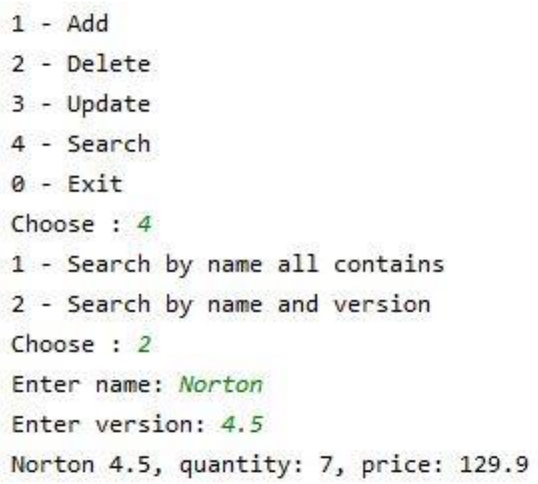
```
test.add(new Package( name: "Adobe Photoshop",  version: "6.0" ,  quantity: 3,  price: 179.90));
test.add(new Package( name: "Adobe Photoshop",  version: "6.2" ,  quantity: 1,  price: 199.90));
test.add(new Package( name: "Norton",  version: "4.5" ,  quantity: 7,  price: 129.90));
test.add(new Package( name: "Norton",  version: "5.5" ,  quantity: 4,  price: 179.90));
test.add(new Package( name: "Adobe Flash",  version: "3.3" ,  quantity: 3,  price: 79.90));
test.add(new Package( name: "Adobe Flash",  version: "4.0" ,  quantity: 3,  price: 109.90));

System.out.println(test.toString());
System.out.println(test.find(new Package( name: "Adobe Flash")));
System.out.println(test.find(new Package( name: "Adobe Flash", version: "4.0")));
System.out.println(test.contains(new Package( name: "Pho")));
int choose;
```



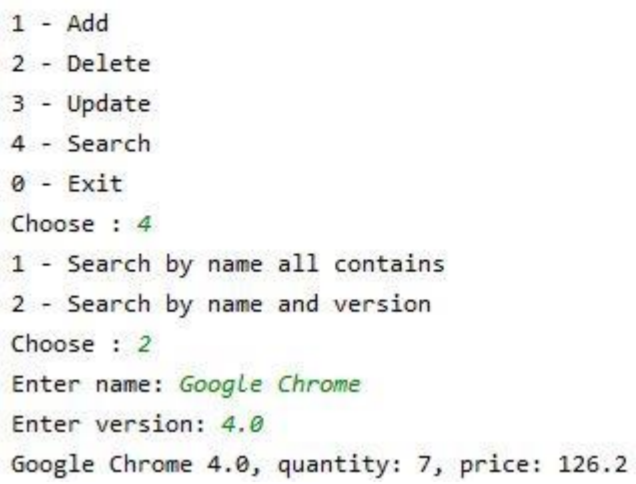
```
Adobe Flash 3.3, quantity: 3, price: 79.9
Adobe Flash 4.0, quantity: 3, price: 109.9
Adobe Photoshop 6.0, quantity: 3, price: 179.9
Adobe Photoshop 6.2, quantity: 1, price: 199.9
Norton 4.5, quantity: 7, price: 129.9
Norton 5.5, quantity: 4, price: 179.9

Adobe Flash 3.3, quantity: 3, price: 79.9
Adobe Flash 4.0, quantity: 3, price: 109.9
Adobe Photoshop 6.0, quantity: 3, price: 179.9
Adobe Photoshop 6.2, quantity: 1, price: 199.9

1 - Administrator
2 - User
Choose : 1
password : admin
1 - Add
2 - Delete
3 - Update
4 - Search
0 - Exit
```



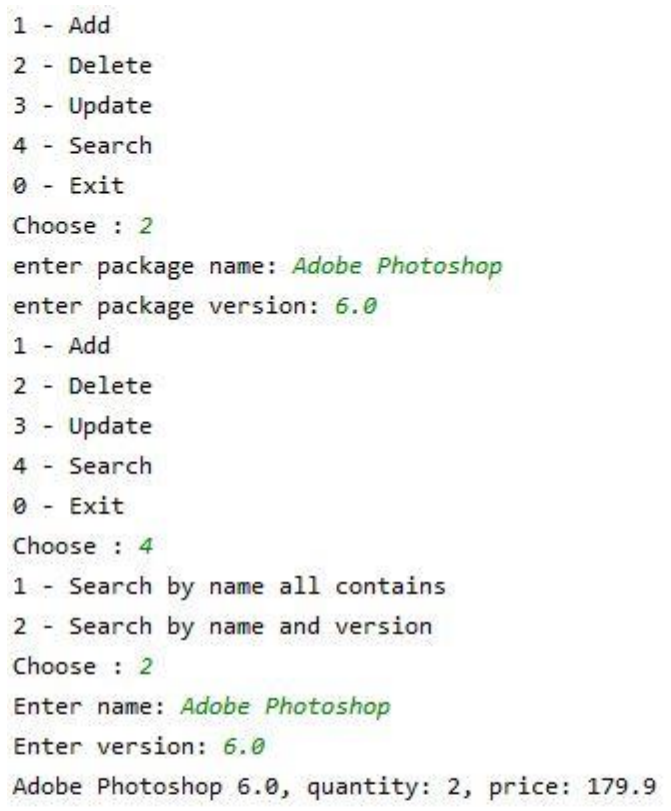
```
Choose : 1
enter package name: Google Chrome
enter package version: 4.0
enter package quantity: 7
enter package price: 126,2
1 - Add
2 - Delete
3 - Update
4 - Search
0 - Exit
Choose : 4
1 - Search by name all contains
2 - Search by name and version
Choose : 1
Enter : Adobe
Adobe Flash 3.3, quantity: 3, price: 79.9
Adobe Flash 4.0, quantity: 3, price: 109.9
Adobe Photoshop 6.0, quantity: 3, price: 179.9
Adobe Photoshop 6.2, quantity: 1, price: 199.9
```



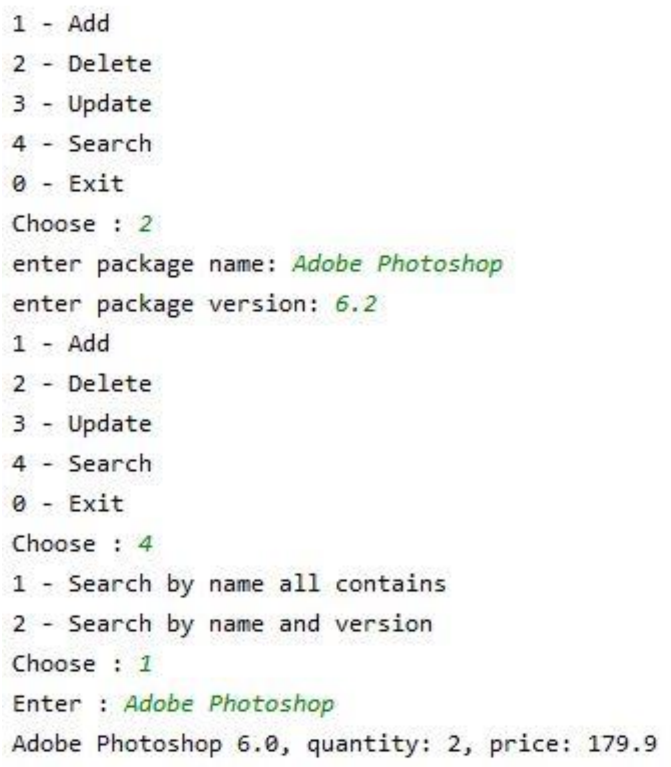
```
1 - Add
2 - Delete
3 - Update
4 - Search
0 - Exit
Choose : 4
1 - Search by name all contains
2 - Search by name and version
Choose : 2
Enter name: Norton
Enter version: 4.5
Norton 4.5, quantity: 7, price: 129.9
```



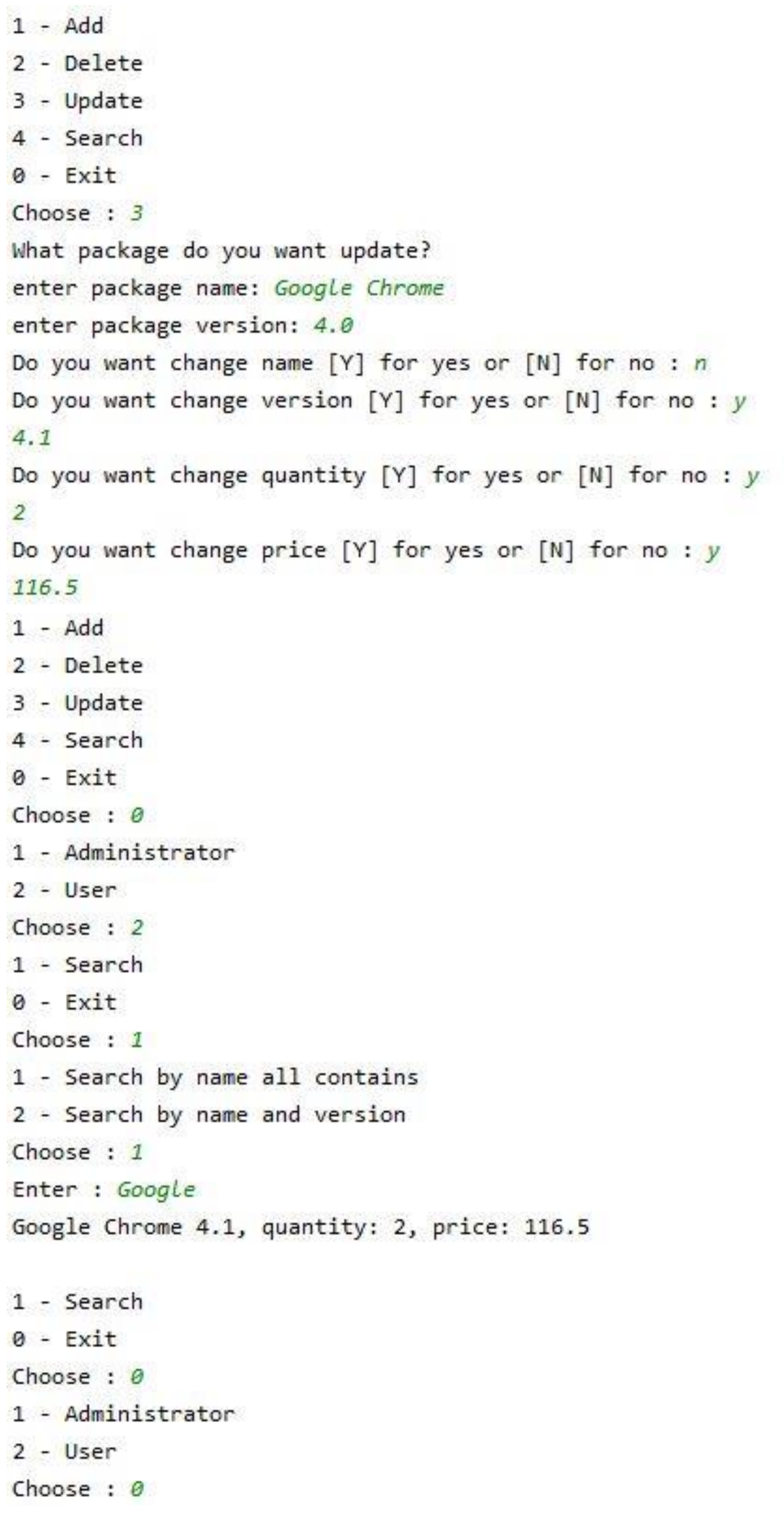
```
1 - Add
2 - Delete
3 - Update
4 - Search
0 - Exit
Choose : 4
1 - Search by name all contains
2 - Search by name and version
Choose : 2
Enter name: Google Chrome
Enter version: 4.0
Google Chrome 4.0, quantity: 7, price: 126.2
```



```
1 - Add
2 - Delete
3 - Update
4 - Search
0 - Exit
Choose : 2
enter package name: Adobe Photoshop
enter package version: 6.0
1 - Add
2 - Delete
3 - Update
4 - Search
0 - Exit
Choose : 4
1 - Search by name all contains
2 - Search by name and version
Choose : 2
Enter name: Adobe Photoshop
Enter version: 6.0
Adobe Photoshop 6.0, quantity: 2, price: 179.9
```



```
1 - Add
2 - Delete
3 - Update
4 - Search
0 - Exit
Choose : 2
enter package name: Adobe Photoshop
enter package version: 6.2
1 - Add
2 - Delete
3 - Update
4 - Search
0 - Exit
Choose : 4
1 - Search by name all contains
2 - Search by name and version
Choose : 1
Enter : Adobe Photoshop
Adobe Photoshop 6.0, quantity: 2, price: 179.9
```



```
1 - Add
2 - Delete
3 - Update
4 - Search
0 - Exit
Choose : 3
What package do you want update?
enter package name: Google Chrome
enter package version: 4.0
Do you want change name [Y] for yes or [N] for no : n
Do you want change version [Y] for yes or [N] for no : y
4.1
Do you want change quantity [Y] for yes or [N] for no : y
2
Do you want change price [Y] for yes or [N] for no : y
116.5
1 - Add
2 - Delete
3 - Update
4 - Search
0 - Exit
Choose : 0
1 - Administrator
2 - User
Choose : 2
1 - Search
0 - Exit
Choose : 1
1 - Search by name all contains
2 - Search by name and version
Choose : 1
Enter : Google
Google Chrome 4.1, quantity: 2, price: 116.5

1 - Search
0 - Exit
Choose : 0
1 - Administrator
2 - User
Choose : 0
```