

Gebze Technical University
Computer Engineering

CSE 222
2017 Spring

HOMEWORK 8 REPORT

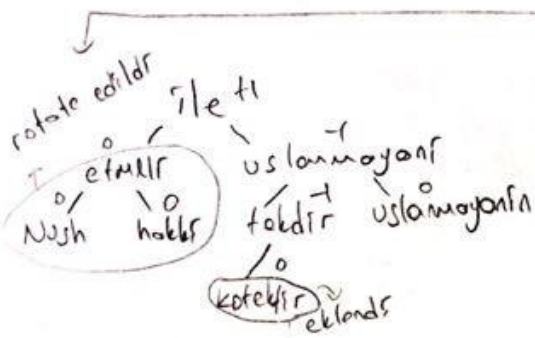
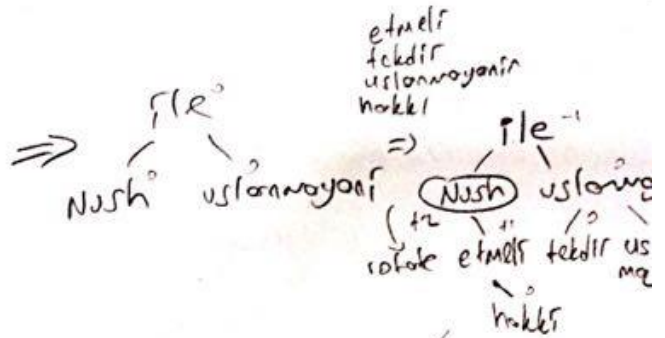
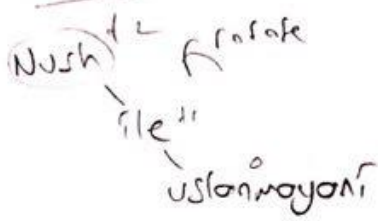
EMRE ÇELİK
141044024

Course Assistant:
ŞEYMA YÜCER

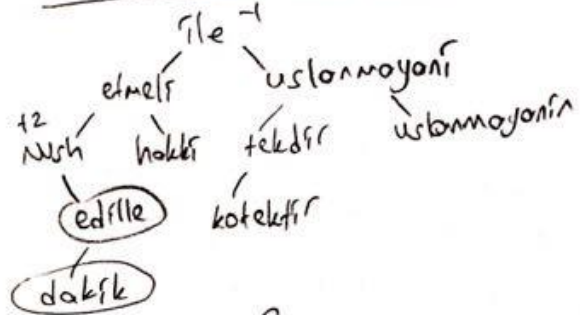
Q1)

Q2)

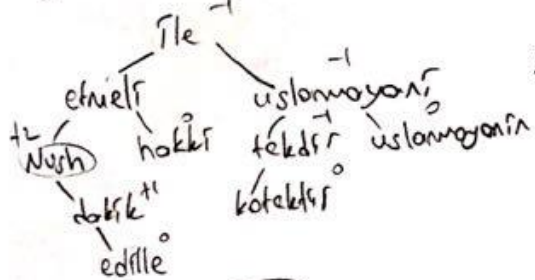
Build



Insert edille dakik



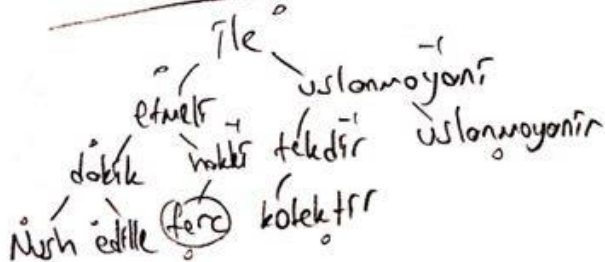
rotate



2. Rotate

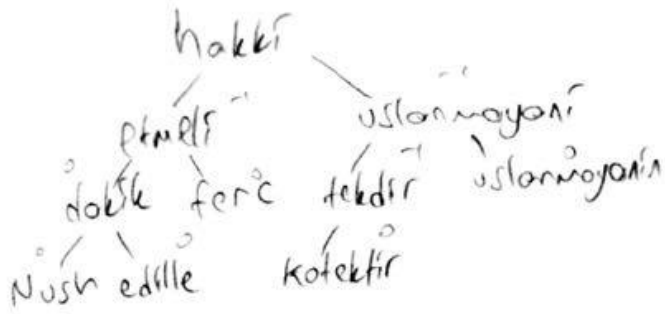


Insert ferc



Q1)

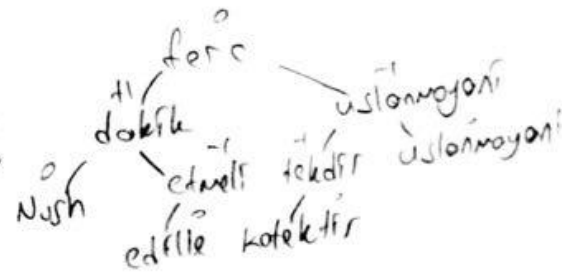
delete ile



delete hakki



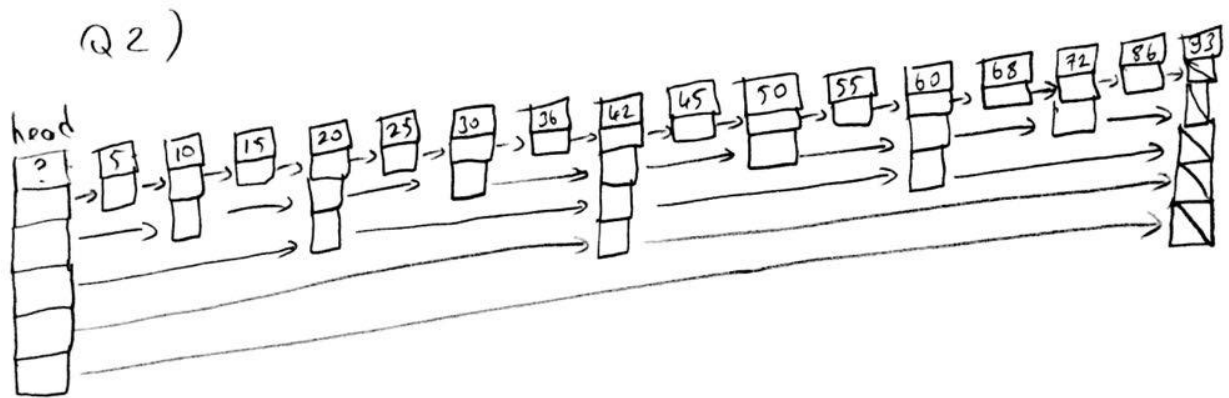
delete \Rightarrow



delete ferc



Q2)



Problem Solutions Approach

İlk partta ağacı build ederken elemanları bst deki gibi kıyaslayıp ağaca eklerken balance degerlerine bakarak herhangi bir nod +2 veya -2 oldugunda , child'ın -1 veya +1 olma durumuna göre left-left(-2,-1) , left-right(-2,+1) , right-left(-2,-1),right-right(+2,+1) rotate işlemlerinden gerekli olanını uyguladım.Eklemeleri bu şekilde yaparken silme işlemlerinde roottan sildim.Silinenin yerine sol ağacın en büyüğünü veya sağ ağacın en küçüğünü seçip roota koyduktan sonra balance kontrolü yapıp gerektiğinde rotate yaptım.İkinci partta ise kitaptaki ideal-skip-list ' te olduğu gibi skip-listi çizdim.16 eleman oldugundan $2^{(5-1)}$ yani max level 5 olacak şekilde çizdim.Üçüncü partta ise kitabın kaynak kodunu kullandım ve işlemleri yanlış yapıyordu.Gerekli düzenlemeleri ve eklemeleri yapıp düzgün çalışır hale getirdim.Ağacı iyi takip edebilmek için aşağıda linkini verdiğim yerden tree'yi düzgün print eden bir kod aldım.

<http://stackoverflow.com/questions/4965335/how-to-print-binary-tree-diagram>

Test Cases

Durum : Ağaçta olan elemanın eklenmeye çalışılması.

Beklenen : Ağaca eklenmemesi

Gerçekleşen : Eleman ağaçta olduğundan bir daha eklenmedi.

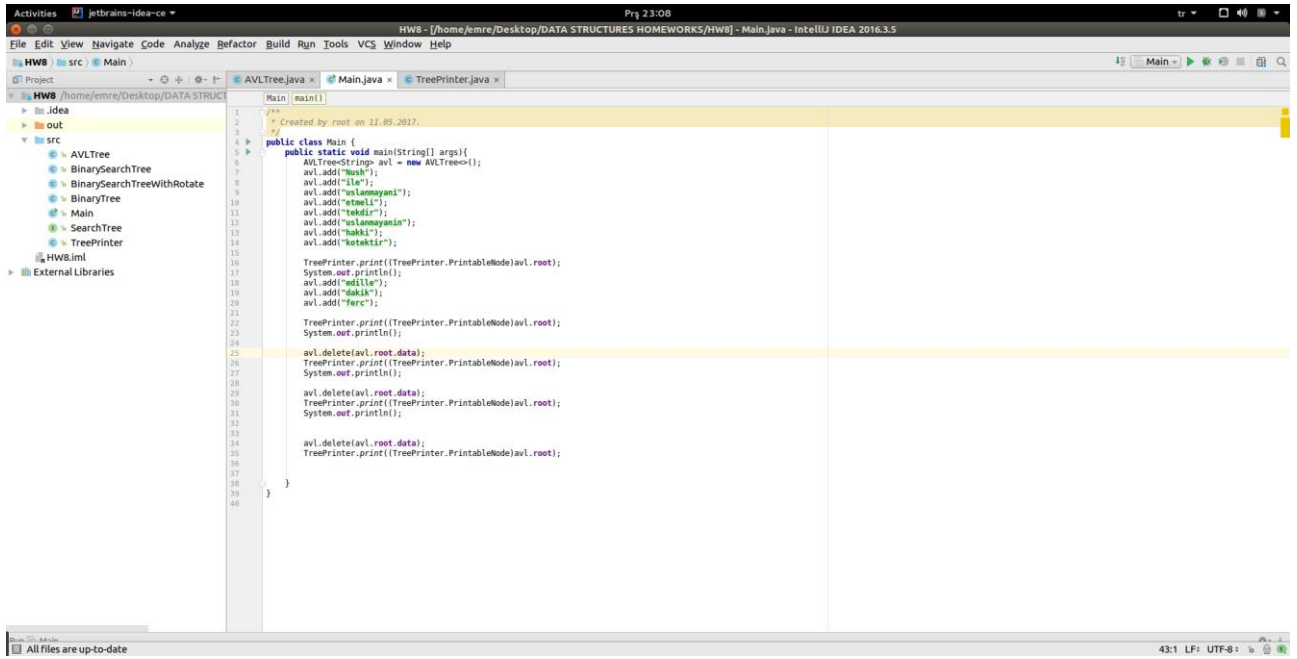
Durum : Parent nodun balance'ının +2 olması , child nodun balance'ının +1 olması

Beklenen : Rotate yapıp balance'ların 0 olması

Gerçekleşen : Parent etrafında left rotate yapıldı ve balance'lar 0 oldu.

Running and Results

İlk soruda istenenleri uyguladım.



```
1  /**
2   * Created by root on 31.05.2017.
3   */
4  public class Main {
5      public static void main(String[] args){
6          AVLTree<String> avl = new AVLTree<>();
7          avl.add("hash");
8          avl.add("ile");
9          avl.add("uslanmayani");
10         avl.add("etmeli");
11         avl.add("tekdir");
12         avl.add("uslanmayani");
13         avl.add("haki");
14         avl.add("kotektir");
15
16         TreePrinter.print((TreePrinter.PrintableNode)avl.root);
17         System.out.println();
18         avl.add("edille");
19         avl.add("dikik");
20         avl.add("ferc");
21
22         TreePrinter.print((TreePrinter.PrintableNode)avl.root);
23         System.out.println();
24
25         avl.delete(avl.root.data);
26         TreePrinter.print((TreePrinter.PrintableNode)avl.root);
27         System.out.println();
28
29         avl.delete(avl.root.data);
30         TreePrinter.print((TreePrinter.PrintableNode)avl.root);
31         System.out.println();
32
33         avl.delete(avl.root.data);
34         TreePrinter.print((TreePrinter.PrintableNode)avl.root);
35
36     }
37 }
38
39
40 }
```

