Gebze Technical University
Computer Engineering

CSE 222
2017 Spring

HOMEWORK 2 REPORT

OSMAN AKKUS
151044055

Course Assistant:Nur Banu Albayrak

1. System Requirements

Give detailed system requirements.

2. Use Case Diagrams

Add use case diagrams if required.

3. Class Diagrams

Add class diagrams.

4. Other Diagrams

Add other diagrams.

5. Problem Solutions Approach

Probleme öncelikle daha önceden ArrayList kullandığım objeleri LinkedList olarak değiştirmek ile başladım. Daha sonrasında bunlar arasında farka bakacak olursak aslında ilk başta gözle görünür bir olmasa da bunların Collection hiyerarşisinden override ettikleri bazı methodların implementasyon farklılıkları aralarında hız farkına sebep olacaktır. Peki bu fonksiyonlar nelerdir ve farkları nedir?

1.Get Function
2.Add Function
3.Remove Function

1.Get function

ArrayList

```
public E get(int index) {
        rangeCheck(index);
        checkForComodification();
        return ArrayList.this.elementData(offset + index);
}
```

LinkedList

```
public E get(int index) {
      checkElementIndex(index);
      return node(index).item;
}
```

ArrayList de performans O(1) yani constant time alırken bu LinkedList de O(n) sürede olur. Çünkü ArrayList index tabanlı çalışıp index deki elemanı direkt olarak bulurken, LinkedList node yardımıyla her elementi aramak durumundadır.

2.Add Function

ArrayList

```
public void add(E e) {
        checkForComodification();

        try {
           int i = cursor;
           ArrayList.this.add(i, e);
           cursor = i + 1;
           lastRet = -1;
           expectedModCount = modCount;
        } catch (IndexOutOfBoundsException ex) {
           throw new ConcurrentModificationException();
        }
}
```

LinkedList

```
public void add(E e) {
        checkForComodification();
        lastReturned = null;
        if (next == null)
           linkLast(e);
        else
           linkBefore(e, next);
        nextIndex++;
        expectedModCount++;
}
```

ArrayList de add methodu O(n) sürede gerçekleşirken bu olay LinkedList de O(1) sürede gerçkleşir. Bunun sebebi ise LinkedList her bir eleman birbirlerine tabiri caizse pointerlarla bağlanmış gibidir. Bu yüzden herhangi bir eleman eklerken yalnız ekleyeceğimiz elemanın pointerının göstereceği yeri o sıradaki elemana işaret ederek yaparız ve herhangi bir kaydırma işlemine gerek duyulmaz ama bu olay ArrayList de herbir elemanın tek tek kaydırılıp yerine koyulması ile mümkün olabilir.

3.Remove Function

ArrayList

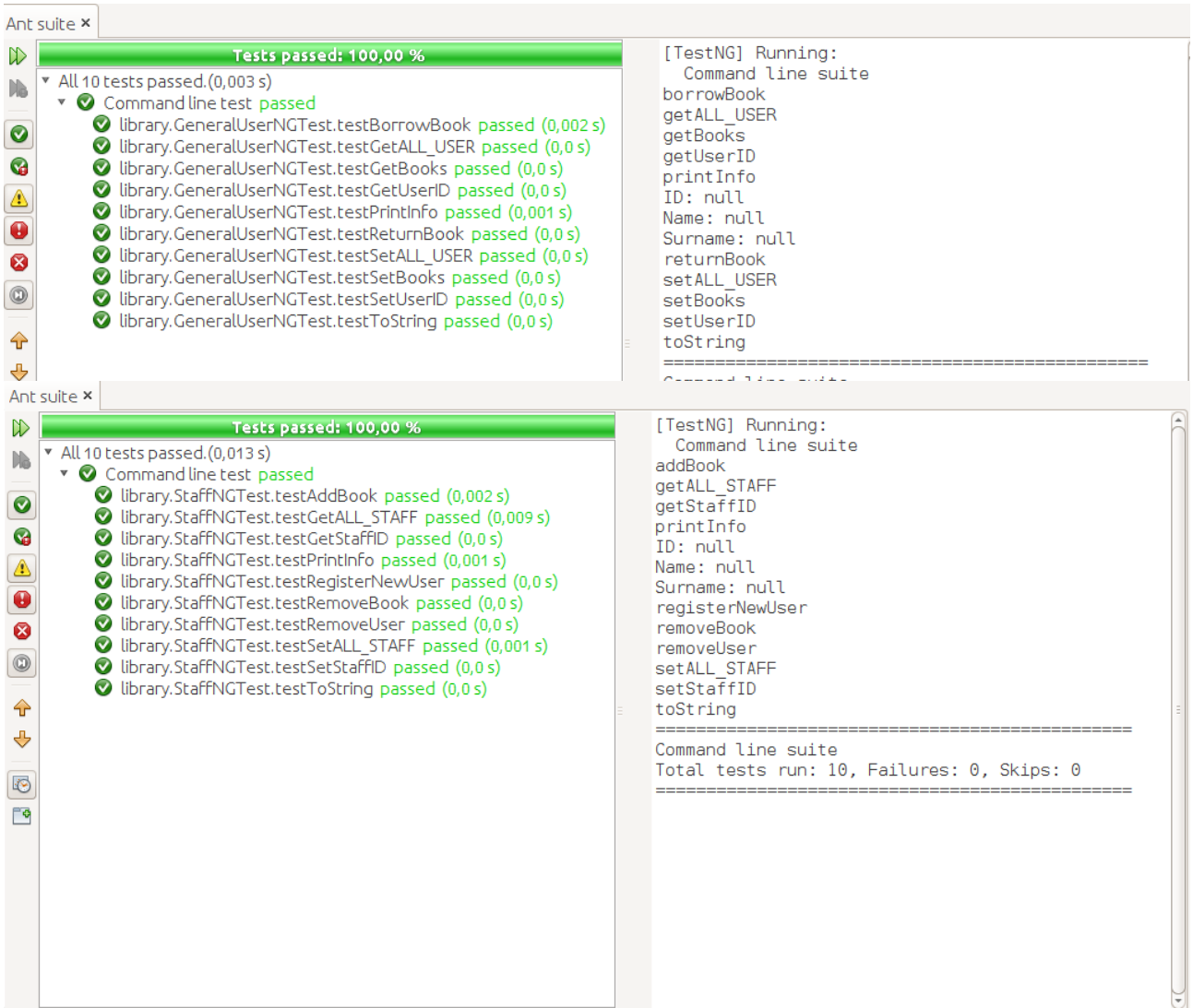
```
public boolean remove(Object o) {
      if (o == null) {
         for (int index = 0; index < size; index++)
            if (elementData[index] == null) {
               fastRemove(index);
               return true;
            }
      } else {
         for (int index = 0; index < size; index++)
            if (o.equals(elementData[index])) {
               fastRemove(index);
               return true;
            }
      }
      return false;
}
```

LinkedList

```
public boolean remove(Object o) {
        if (o == null) {
            for (Node<E> x = first; x != null; x = x.next) {
                if (x.item == null) {
                    unlink(x);
                    return true;
                }
            }
        } else {
            for (Node<E> x = first; x != null; x = x.next) {
                if (o.equals(x.item)) {
                    unlink(x);
                    return true;
                }
            }
        }
        return false;
}
```

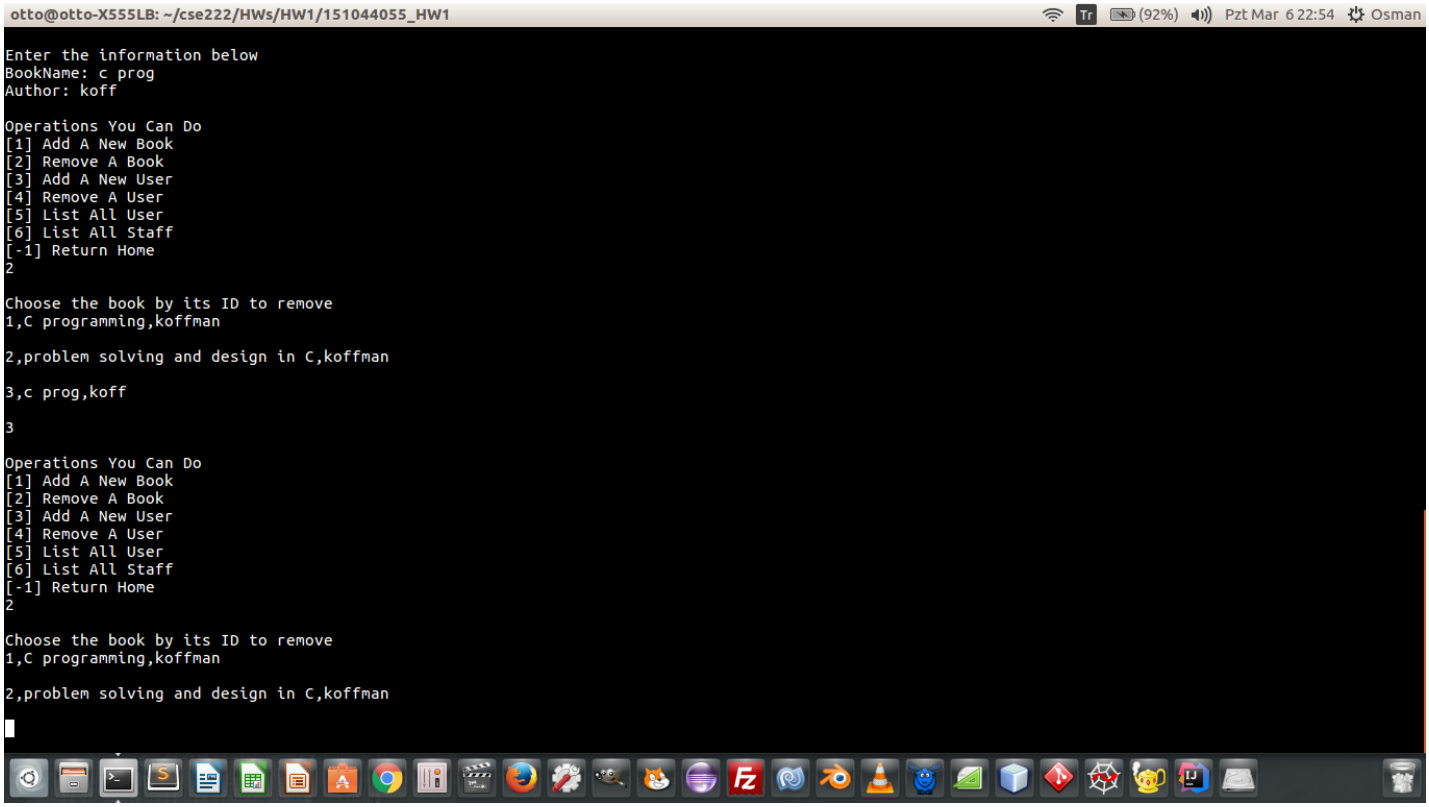
Burada da aynı sekilde LinkedList ArrayList e göre daha hızlı çalışacaktır. ArrayList O(n^2) ile çalışacak iken LinkedList O(n) performansıyla çalışacaklardır.Sebebi ise yukarda belirttiğimiz add fonksiyonun temel sebebidir.ArrayList remove işlemini yaptıktan sonra kaydırma yapcağı için daha yavaşdır.

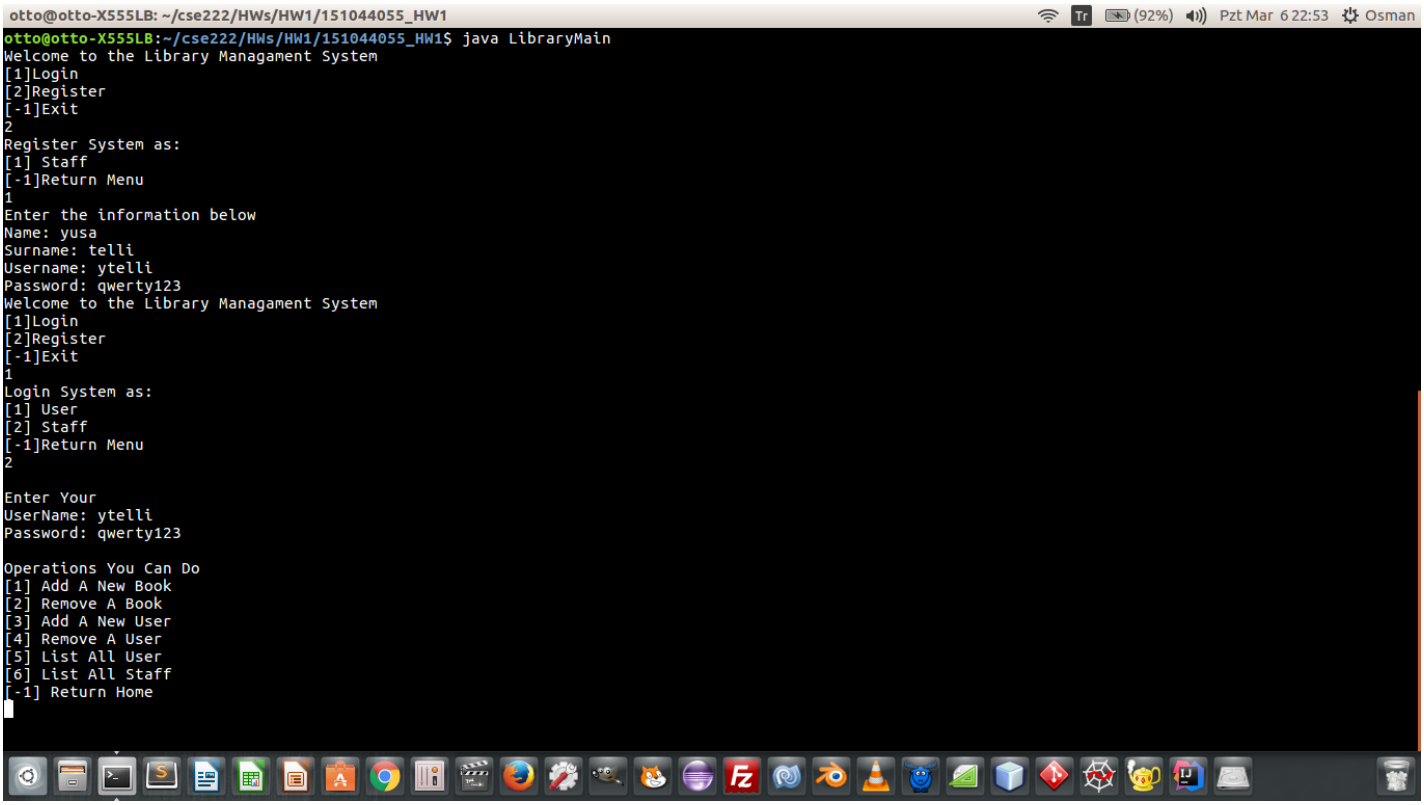
6. Test Cases

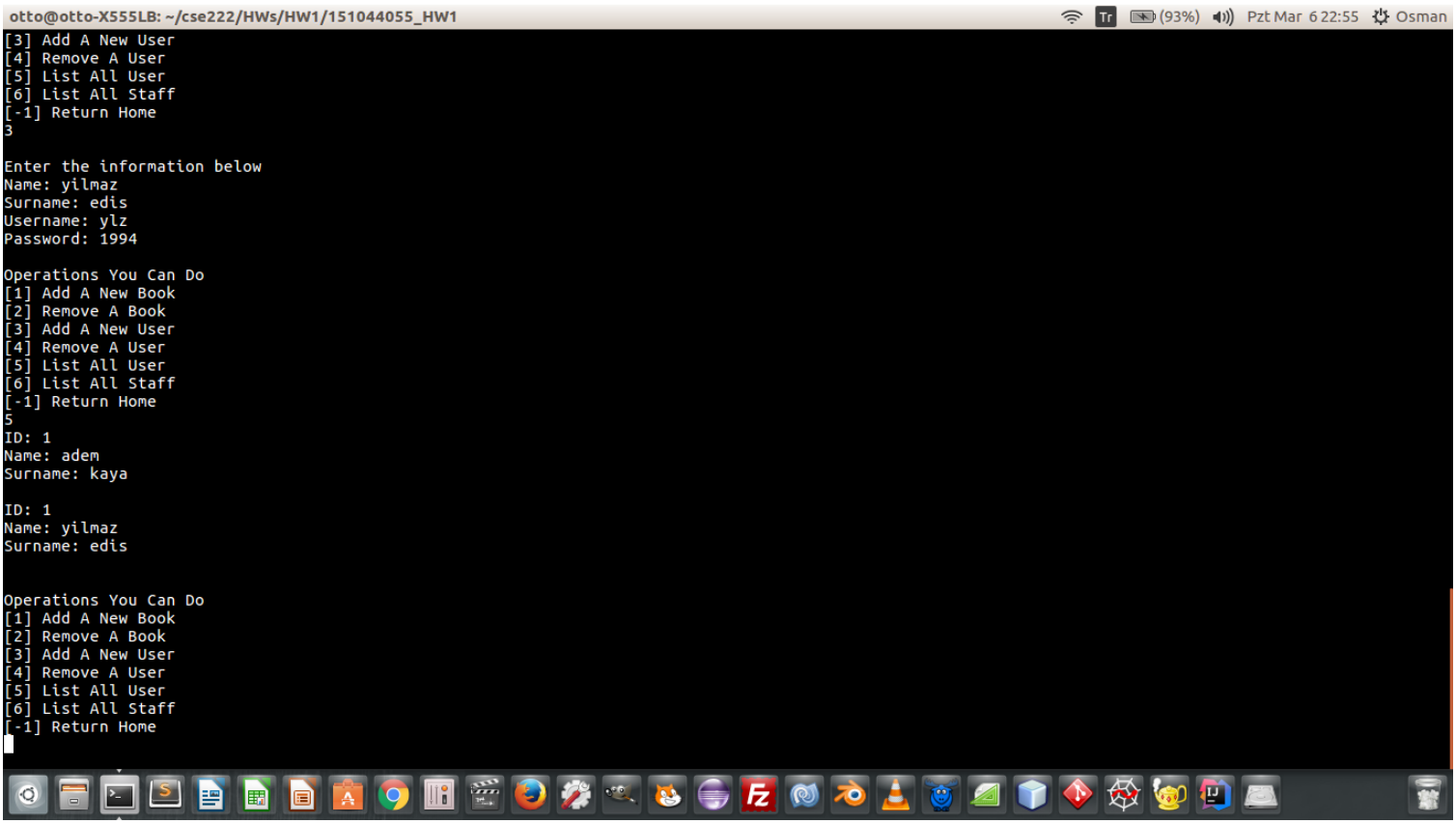


7. Running and Results

There are some running results screenshots


otto@otto-X555LB: ~/cse222/HWs/HW1/151044055_HW1
Pzt Mar 6 22:54
Osman
Enter the information below
BookName: c prog
Author: koff
Operations You Can Do
[1] Add A New Book
[2] Remove A Book
[3] Add A New User
[4] Remove A User
[5] List All User
[6] List All Staff
[-1] Return Home
2
Choose the book by its ID to remove
1,C programming,koffman
2,problem solving and design in C,koffman
3,c prog,koff
3
Operations You Can Do
[1] Add A New Book
[2] Remove A Book
[3] Add A New User
[4] Remove A User
[5] List All User
[6] List All Staff
[-1] Return Home
2
Choose the book by its ID to remove
1,C programming,koffman
2,problem solving and design in C,koffman


otto@otto-X555LB: ~/cse222/HWs/HW1/151044055_HW1
Pzt Mar 6 22:53
Osman
otto@otto-X555LB:~/cse222/HWs/HW1/151044055_HW1$ java LibraryMain
Welcome to the Library Managament System
[1]Login
[2]Register
[-1]Exit
2
Register System as:
[1] Staff
[-1]Return Menu
1
Enter the information below
Name: yusa
Surname: telli
Username: ytelli
Password: qwerty123
Welcome to the Library Managament System
[1]Login
[2]Register
[-1]Exit
1
Login System as:
[1] User
[2] Staff
[-1]Return Menu
2
Enter Your
UserName: ytelli
Password: qwerty123
Operations You Can Do
[1] Add A New Book
[2] Remove A Book
[3] Add A New User
[4] Remove A User
[5] List All User
[6] List All Staff
[-1] Return Home


otto@otto-X555LB: ~/cse222/HWs/HW1/151044055_HW1
(93%)
Pzt Mar 6 22:55
Osman
[3] Add A New User
[4] Remove A User
[5] List All User
[6] List All Staff
[-1] Return Home
3
Enter the information below
Name: yilmaz
Surname: edis
Username: ylz
Password: 1994
Operations You Can Do
[1] Add A New Book
[2] Remove A Book
[3] Add A New User
[4] Remove A User
[5] List All User
[6] List All Staff
[-1] Return Home
5
ID: 1
Name: adem
Surname: kaya
ID: 1
Name: yilmaz
Surname: edis
Operations You Can Do
[1] Add A New Book
[2] Remove A Book
[3] Add A New User
[4] Remove A User
[5] List All User
[6] List All Staff
[-1] Return Home

Q1)

1) `for(int i=0; i<n-1; i++) {`
`for(j=i+1; j<n; j++) {`
3 simple statements
`}`
`}`

$T(n) = 3(n-1) + 3(n-2) + \dots + 1$

$= 3(n-1 + n-2 + \dots + 1)$

Sum of parentheses $\frac{n \times (n-1)}{2}$

$T(n) = 1.5n^2 - 1.5n$

to conclude $T(n) = \Theta(n^2)$

2) `public static int length(String str) {`
`if (str == null || str.equals("")) return 0;`
`else return 1 + length(str.substring(1)); }` $1 + T(n-1)$

$T(n) = 2 + T(n-1) \rightarrow T(n) = \Theta(n)$

Q2)

1) This function is an ordinary sorting algorithm.

2) Best case : The best case is being already sorted.

inner loop $\rightarrow T_B(n) = 1+n+1+n+n+1 = 3n+3 = \Theta(n)$

outter loop $\rightarrow T_B(n) = 2n(3n+3) = 6n^2+6n = \Theta(n^2)$

worst case : The worst case is being decreasing order.

inner loop $\rightarrow T_W(n) = 1+n+1+n+5+n = 3n+7 = \Theta(n)$

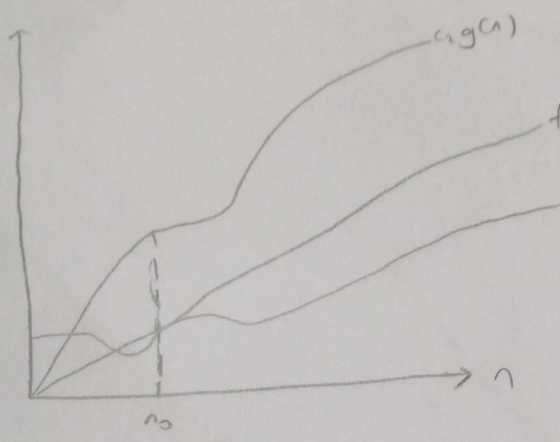
outer loop $\rightarrow T_W(n) = 2n(3n+7) = 6n^2+14n = \Theta(n^2)$

$T(n) = \Theta(n)$

Q3) Definition of Θ

and mst.edu/mercol/.pdf

$\Theta(g(n)) = \{f(n)$: there exist positive constants c_1, c_2 and n_0 such that $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \ge n_0\}$



$\Theta(g(n))$ is the set of functions with the same order of growth

$T(N) = \Theta(h(N))$ if and only if $T(N) = O(h(N))$ and $T(N) = \Omega(h(N))$

Ex: $\frac{n^2 - n}{2} = \Theta(n^2)$

$- \frac{n^2}{2} - \frac{n}{2} \le \frac{n^2}{2} \quad \forall n \ge 0 \Rightarrow c_2 = \frac{1}{2}$

$- \frac{1}{2}n^2 - \frac{1}{2}n \ge \frac{1}{2}n^2 - \frac{1}{2}n * \frac{1}{2}n \ (\forall n \ge 2) = \frac{1}{4}n^2 \Rightarrow c_1 = \frac{1}{4}$

$\Rightarrow$ only holds for: $n \le 1/c_1$

- $n \ne \Theta(n^2)$: $c_1 n^2 \le n \le c_2 n^2$

- $6n^3 \ne \Theta(n^2)$: $c_1 n^2 \le 6n^3 \le c_2 n^2$

$\Rightarrow$ only hold for: $n \le \frac{c_2}{6}$

- $n \ne \Theta(\log n)$: $c_1 \log n \le n \le c_2 \log n$

$\Rightarrow c_2 \ge n/\log n, \ \forall n \ge n_0$ - impossible

Q4) resources: http://www.atmbabu.wordpress.com/2014/01/08/

1) Pseudo - code

```
INSERTION - SORT (A, n):
    if   n > 1 :
        then   INSERTION - SORT (A, n-1)
          key = A[n]
            i = n-1
        while   i > 0      A[i] > key :
                A[i+1] = A[i]
                     i = i-1
                A[i+1] = key
```

2) Best case and worst case

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1, \rightarrow \text{best case} \\ T(n-1) + \Theta(n) & \text{if } n > 1 \rightarrow \text{worst case} \end{cases}$$

3) $T(n) = T(n-1) + (n-1)$

$= T(n-2) + (n-2) + (n-1);$

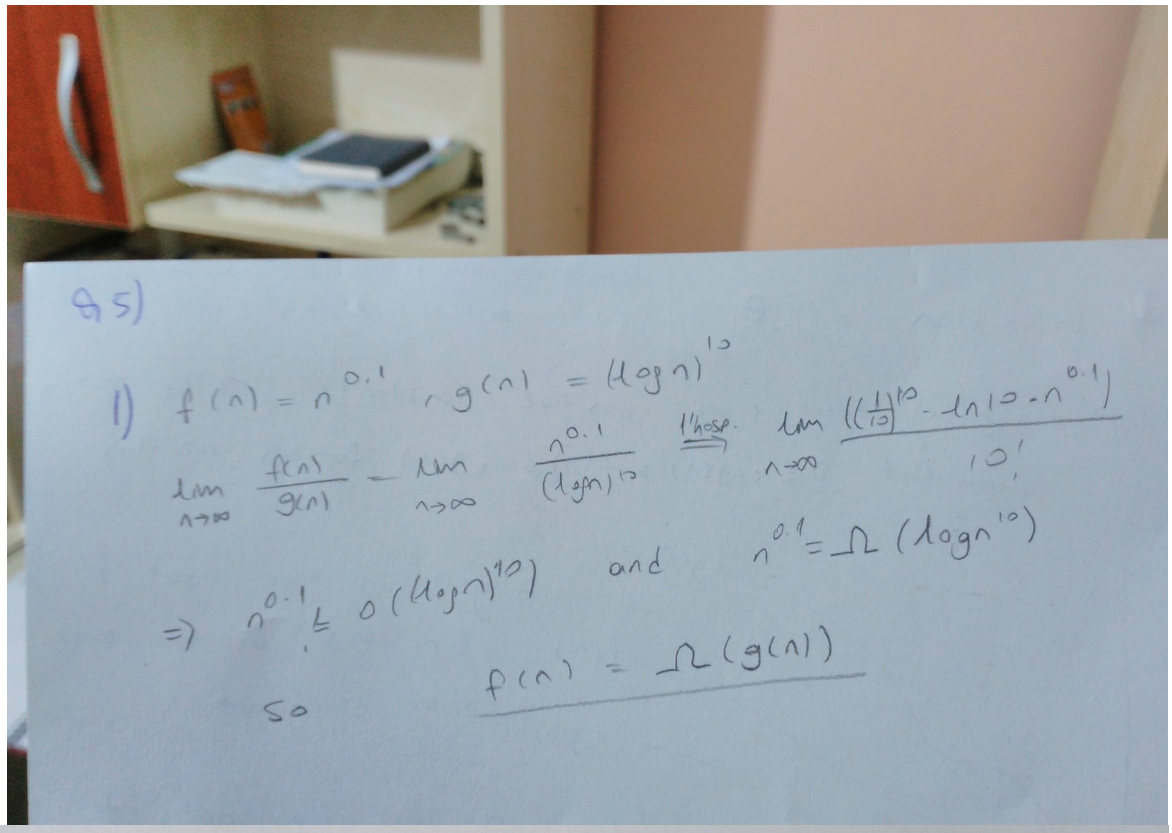
$\vdots$

$= T(2) + 2 + 3 + \cdots (n-2) + (n-1);$

$= T(1) + 1 + 2 + \cdots (n-2) + (n-1);$

$= 1 + 2 + 3 + 4 \quad \cdots \quad (n-2) + (n-1);$

$= \Theta(n^2)$



Q5)

1) $f(n) = n^{0.1}$, $g(n) = (\log n)^{10}$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n^{0.1}}{(\log n)^{10}} \overset{\text{l'hosp.}}{\Longrightarrow} \lim_{n\to\infty} \frac{\left(\left(\frac{1}{10}\right)^{10} \cdot \ln 10 \cdot n^{0.1}\right)}{10!}$$

$\Rightarrow$ $n^{0.1} \neq O((\log n)^{10})$ and $n^{0.1} = \Omega(\log n^{10})$

So $f(n) = \Omega(g(n))$

2) $f(n) = n!$, $g(n) = 2^n$

$n \geq 4$

$c = 100$

$n!$ daha hızlı büyür

$f(n) = \Omega(g(n))$

3) $f(n) = (\log n)^{\log n}$, $g(n) = 2^{(\log_2 n)^2}$

$\log n \log(\log n)$, $(\log_2 n)^2 \log 2 \rightarrow 2 \cdot \log(\log_2 n) \log 2$

Bu sayede $g(n)$, $f(n)$'den daha hızlı büyüyecektir.

$f(n) = O(g(n))$