$3^3 = 3 \cdot 3 \cdot 3 = 27$

$x^x = x_1 \cdot x_2 \cdots x_{n-1} x_n$

# CSI Assignmen 1

Q2

a) $f(n) = \frac{1}{100}n^2 + 100n + 90$ is $O(n^2)$

$\frac{1}{100}n^2 \le c \cdot n^2$ when $c = \frac{1}{100}$ for all $n \ge 1$

$100n \le c \cdot n^2$ when $c = 100$ for all $n \ge$ ~~100~~

$90 \le c \cdot n^2$ when $c = 90$ for all $n \ge 1\cancel{0}$

$\rightarrow \quad c = \frac{1}{100} + 100 + 90 \qquad n_0 = 1$

$= \frac{1}{100} + \frac{10000}{100} + \frac{9000}{100}$

$= \frac{19001}{100}$

~~190.01~~

$\boxed{\text{True}}$ $\boxed{c = \frac{19001}{100},\ n_0 = 1}$

$\because \frac{1}{100}n^2 + 100n + 90 \le \frac{19001}{100}n^2$ for all $n \ge 1$

$\therefore F(n) \in O(n^2) \iff$ there exist $c, n_0$ such that $f(n) \le c\,g(n^2)$ for all $n \ge n_0$

b)

$n\log n + 100\log n$ is $\Omega(\sqrt{n})$

$f(n) = n\log n + 100\log n$

$\rightarrow n\log n \ge c \cdot \sqrt{n}$ when $c = 1$ for all $n \ge$ ~~0~~ $3.\cancel{x}55$

~~$n\log n \ge \sqrt{n}$~~

~~$n\log n \ge n^{-1}$~~

~~$\log n \ge \frac{1}{n^2}$~~

~~$\log n \ge (n^2)^{-1}$~~

$n\log n \ge \sqrt{n}$

$n\log n \ge \frac{n}{\sqrt{n}}$

$\log n \ge \frac{1}{\sqrt{n}}$

~~$n = \sqrt{n} \cdot \sqrt{n}$~~

$\sqrt{n} = \frac{n}{\sqrt{n}}$

$n\log n \ge \sqrt{n}$

$\sqrt{n}\log n \ge 1$

$\log n^{\sqrt{n}} \ge 1$

$n^{\sqrt{n}} \ge 10^1$

$n\log n \ge \sqrt{n}$ $\quad t^2 = n$

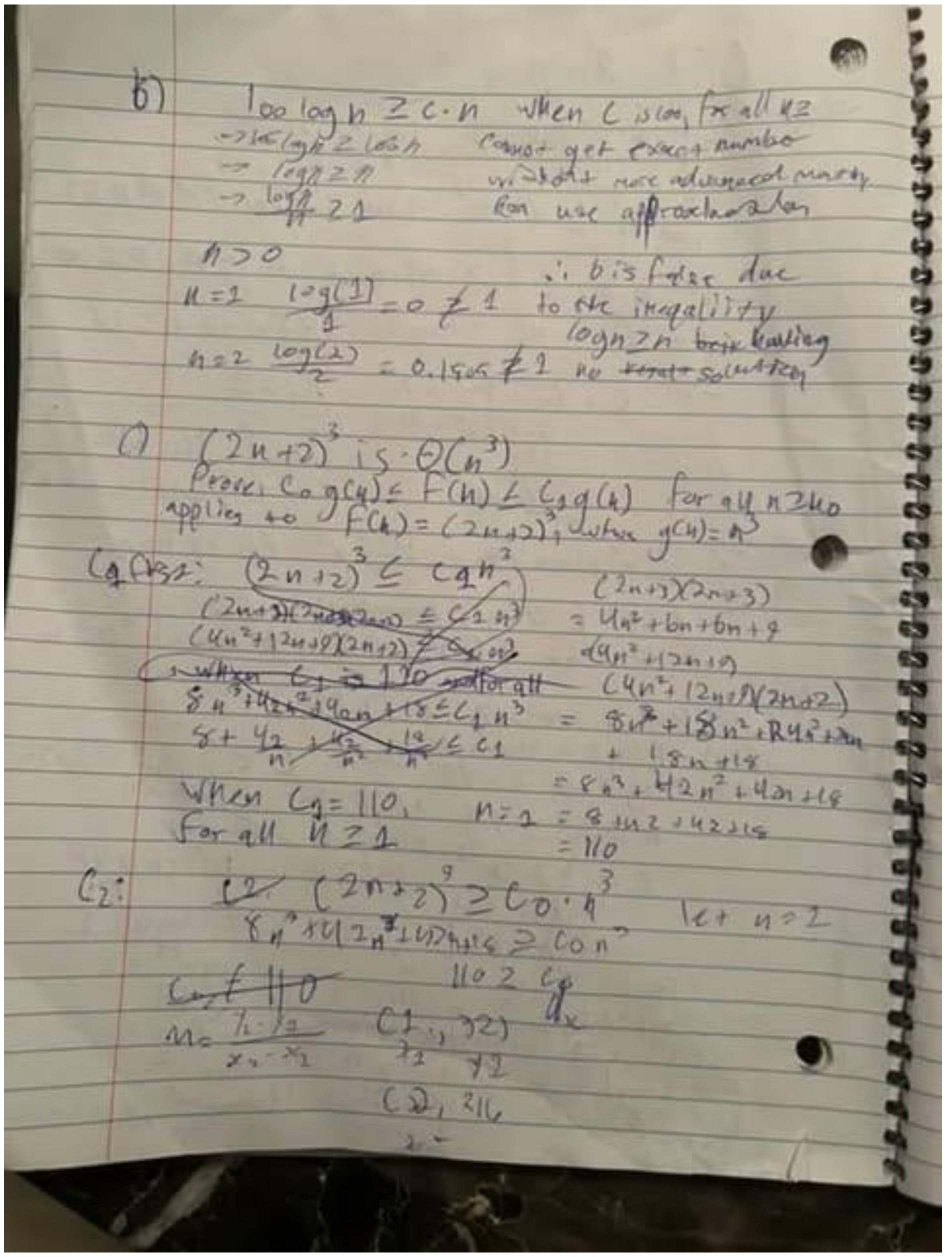
$\rightarrow t^2\log t^2 = t \quad t = \sqrt{n}$

$\rightarrow 2t\log t \ge 1$ ~~Using approximation~~

$\rightarrow t\log t \ge \frac{1}{2}$ equality is impossible to solve without advanced mathematics, but can use approximation

$t = 3.1 = \sqrt{n}$ [illegible]

$1.83 \le \sqrt{3}$ [illegible]



b) $100 \log n \geq c \cdot n$ when $c$ is 100, for all $n \geq$

$\rightarrow 100 \log n \geq 100 n$ cannot get exact number

$\rightarrow \log n \geq n$ without more advanced math,

$\rightarrow \frac{\log n}{n} \geq 1$ can use approximation

$n > 0$

∴ b is false due to the inequality $\log n \geq n$ ~~being~~ having no ~~[illegible]~~ solution

$n=1$ $\frac{\log(1)}{1} = 0 \not\geq 1$

$n=2$ $\frac{\log(2)}{2} = 0.1505 \not\geq 1$

() $(2n+2)^3$ is $\Theta(n^3)$

Prove: $c_0 g(n) \leq f(n) \leq c_1 g(n)$ for all $n \geq n_0$

applies to $f(n) = (2n+2)^3$, where $g(n) = n^3$

$c_1$ first: $(2n+2)^3 \leq c_1 n^3$

~~$(2n+2)(2n+2)(2n+2) \leq c_1 n^3$~~

~~$(4n^2+12n+9)(2n+2) \leq c_1 n^3$~~

~~when $c_1$ is 110 for all~~

~~$8n^3+42n^2+40n+18 \leq c_1 n^3$~~

~~$8 + \frac{42}{n} + \frac{42}{n^2} + \frac{18}{n^3} \leq c_1$~~

When $c_1 = 110$, for all $n \geq 1$

$(2n+3)(2n+3)$
$= 4n^2+6n+6n+9$
$= (4n^2+12n+9)$
$(4n^2+12n+9)(2n+2)$
$= 8n^3 + 18n^2 + 24n^2 + 24n + 18n + 18$
$= 8n^3 + 42n^2 + 40n + 18$
$n=1$ $= 8+42+42+18$
$= 110$

$c_2$: ~~$c_2$~~ $(2n+2)^3 \geq c_0 \cdot n^3$  let $n=2$

$8n^3+42n^2+42n+18 \geq c_0 n^3$

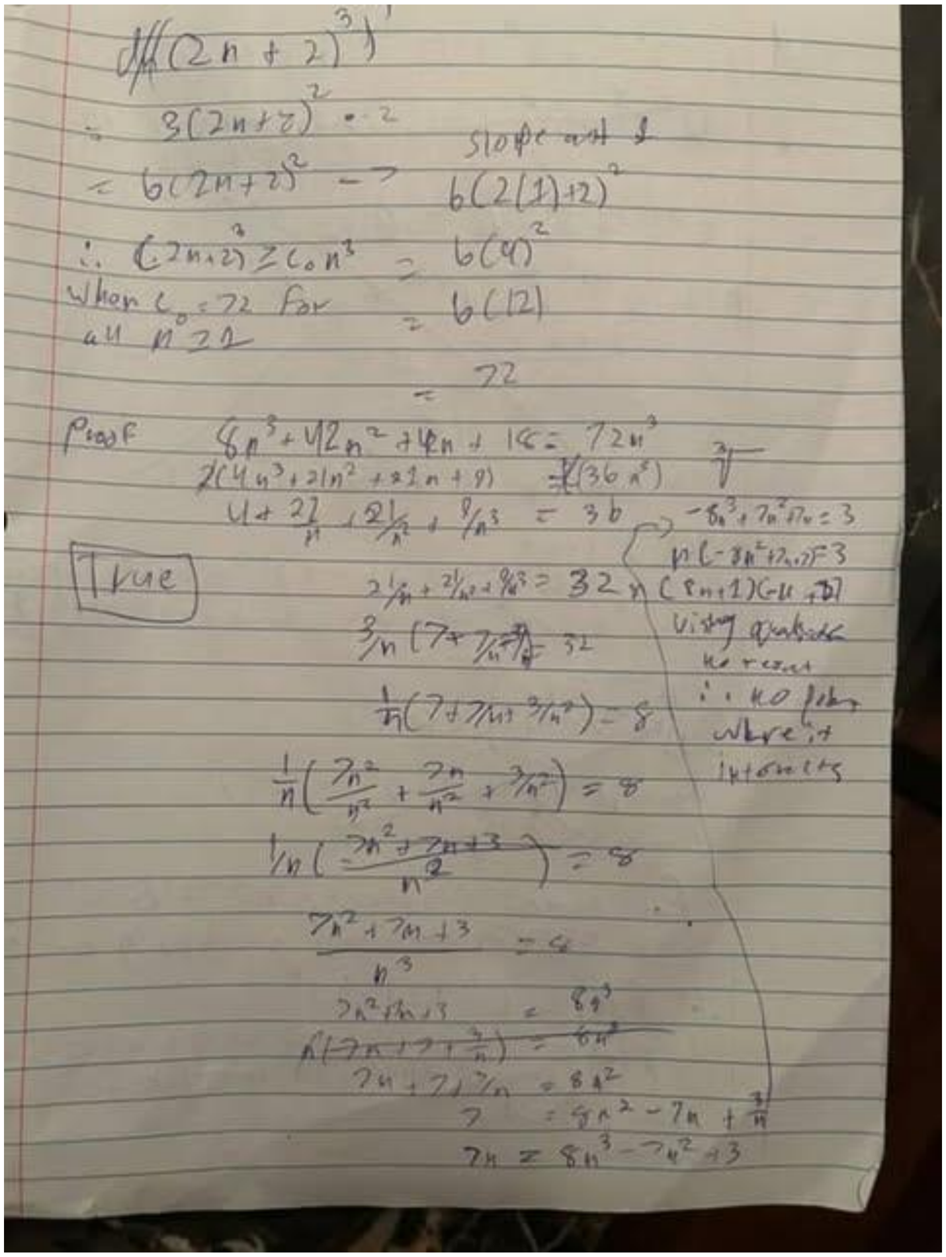
~~$c_0 \neq 110$~~  $110 \geq c_0$

$m = \frac{y_1 - y_2}{x_1 - x_2}$  $(1, 72)$ $d_x$

$x_1$ $y_2$

$(2, 216$

$2$



$\frac{d}{dn}((2n+2)^3)$

$= 3(2n+2)^2 \cdot 2$

$= 6(2n+2)^2 \longrightarrow$ slope at 1 $6(2(1)+2)^2$

$= 6(4)^2$

$= 6(12)$

$= 72$

$\therefore (2n+2)^3 \geq c_0 n^3$ when $c_0 = 72$ for all $n \geq 1$

Proof $8n^3 + 42n^2 + 42n + 16 = 72n^3$

$\cancel{2}(4n^3 + 21n^2 + 21n + 8) = \cancel{2}(36n^3)$

$4 + \frac{21}{n} + \frac{21}{n^2} + \frac{8}{n^3} = 36$

$\boxed{\text{True}}$

$\frac{21}{n} + \frac{21}{n^2} + \frac{8}{n^3} = 32$

$\frac{3}{n}(7 + \frac{7}{n} + \frac{3}{n^2}) = 32$

$\frac{1}{n}(7 + \frac{7}{n} + \frac{3}{n^2}) = 8$

$\frac{1}{n}(\frac{7n^2}{n^2} + \frac{7n}{n^2} + \frac{3}{n^2}) = 8$

$\frac{1}{n}(\frac{7n^2 + 7n + 3}{n^2}) = 8$

$\frac{7n^2 + 7n + 3}{n^3} = 8$

$7n^2 + 7n + 3 = 8n^3$

~~$n(7n + 7 + \frac{3}{n}) = 8n^3$~~

$7n + 7 + \frac{3}{n} = 8n^2$

$7 = 8n^2 - 7n + \frac{3}{n}$

$7n = 8n^3 - 7n^2 + 3$

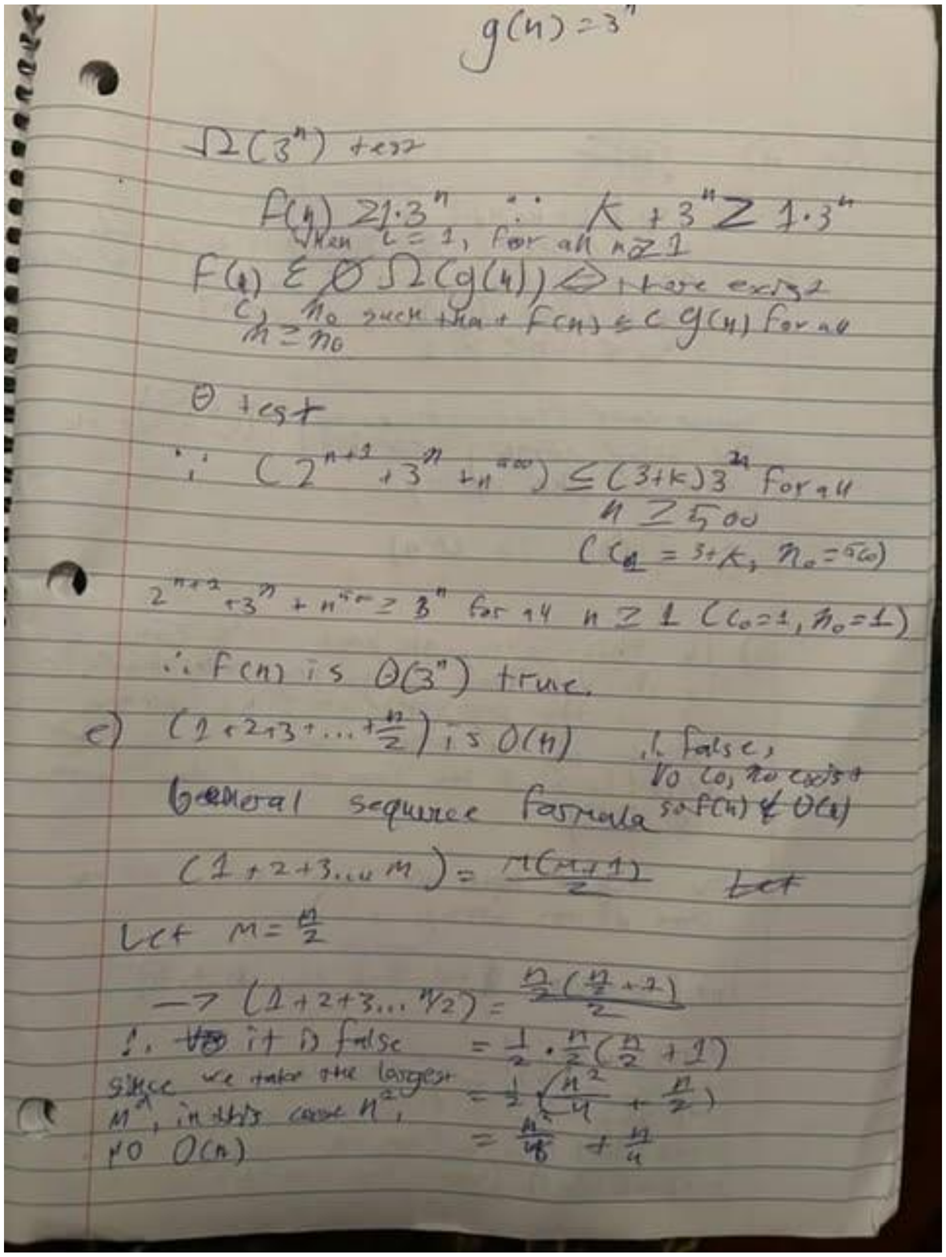
$-8n^3 + 7n^2 + 7n = 3$

$n(-8n^2 + 7n + 7) = 3$

$n(8n+1)(-n+7)$

using quadratic no reent

$\therefore$ no place where it intersects



$g(n) = 3^n$

$\Omega(3^n)$ test

$f(n) \geq 1 \cdot 3^n \quad \therefore \quad k + 3^n \geq 1 \cdot 3^n$
when $c = 1$, for all $n \geq 1$

$f(n) \in \Omega(g(n)) \Leftrightarrow$ there exist $c, n_0$ such that $f(n) \leq c\, g(n)$ for all $n \geq n_0$

$\Theta$ test

$\therefore \; (2^{n+1} + 3^n + n^{500}) \leq (3+k)3^n$ for all $n \geq 500$
$(c_1 = 3+k,\ n_0 = 500)$

$2^{n+1} + 3^n + n^{500} \geq 3^n$ for all $n \geq 1$ $(c_0 = 1, n_0 = 1)$

$\therefore f(n)$ is $\Theta(3^n)$ true.

e) $(1+2+3+\ldots+\frac{n}{2})$ is $O(n)$ $\therefore$ false, no $c_0, n_0$ exist so $f(n) \notin O(n)$

General sequence formula

$(1+2+3\ldots m) = \frac{m(m+1)}{2}$ ~~Let~~

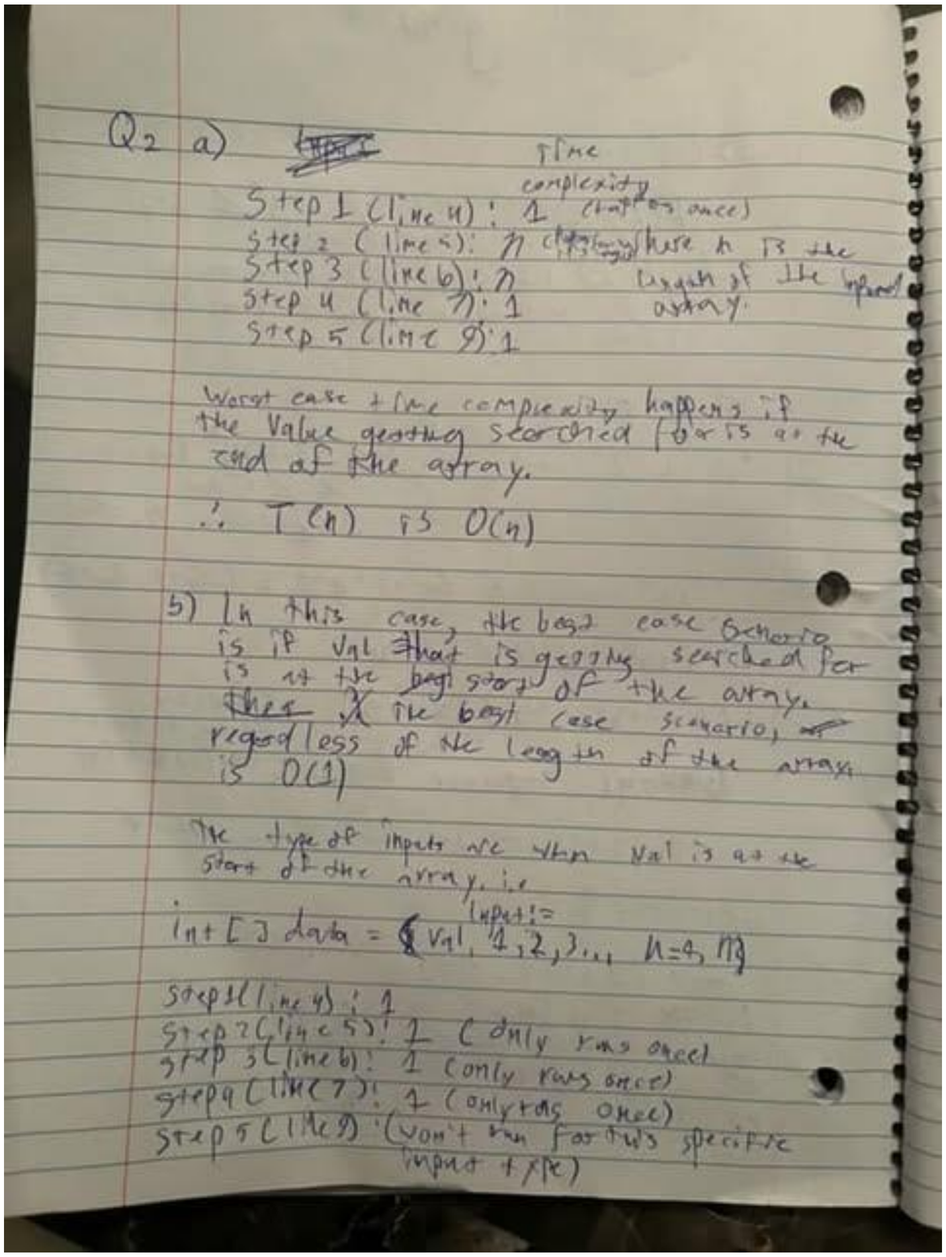
Let $m = \frac{n}{2}$

$\rightarrow (1+2+3\ldots n/2) = \frac{\frac{n}{2}(\frac{n}{2}+1)}{2}$

$\therefore$ ~~[illegible]~~ it is false
since we take the largest $n^x$, in this case $n^2$, no $O(n)$

$= \frac{1}{2} \cdot \frac{n}{2}(\frac{n}{2}+1)$
$= \frac{1}{2}(\frac{n^2}{4} + \frac{n}{2})$
$= \frac{n^2}{8} + \frac{n}{4}$

Questions 2



Q2 a) ~~[illegible]~~

Time complexity

Step 1 (line 4): 1 (runs once)
Step 2 (line 5): $n$ ~~[illegible]~~ where $n$ is the length of the input array.
Step 3 (line 6): $n$
Step 4 (line 7): 1
Step 5 (line 9): 1

Worst case time complexity happens if the Value getting searched for is at the end of the array.

∴ $T(n)$ is $O(n)$

b) In this case, the best case scenario is if val ~~that~~ is getting searched for is at the ~~beg~~ start of the array. ~~then~~ ~~X~~ the best case scenario, regardless of the length of the array is $O(1)$

The type of inputs are when val is at the start of the array, i.e

input:=
int [] data = {val, 1, 2, 3..., n-1, n}

step1 (line 4): 1
Step 2 (line 5): 1 (only runs once)
step 3 (line 6): 1 (only runs once)
step4 (line 7): 1 (only runs once)
Step 5 (line 9): (won't run for this specific input type)

Question 3:

public class ForgetfullStack<E> implements Stack<E>{

public class Node<E>;

private E object;

private Node<E> next;

private Node<E> prev;

```
public Node(){

this(null,null,this);

}


public Node(E obj, Node<E> n, Node<E> p){

object = obj;

next = n;

prev = p;

}


public E getObject() {

return object;

}


public Node<E> getNext() {

return next;

}


public Node<E> getPrev() {

return prev;

}


public void setObject(E newObj) {

object = newObj;

}
```

```
public void setNext(Node<E> newNext) {

next = newNext;

}


public void setPrev(Node<E> newPrev) {

next = newPrev;

}


int sz = 0;

Node<String> head;

Node<String> dummy;


head.setPrev(dummy);

head.setNext(dummy);


dummy.setPrev(head);

dummy.setNext(head);


public int size(){

return sz;

}


public boolean isEmpty(){

return(head.getNext() == dummy);

}
```

```
public E top(){

if(isEmpty()){

return null;

}

else{

return head.getNext;

}

}


public void push(E Object){

Node<String> temp = new Node<>(Object, null);

if(isEmpty()){

head.setNext(temp);

dummy.setPrev(temp);

temp.setNext(dummy);

sz++;

}

else{

Node<String> current = head.getNext;

temp.setNext(current);

current.setPrev(temp);

head.setNext(temp);

sz ++;

}
}


public E pop(){
if(isEmpty()){
return null;
}
else{
Node<String> current = head.getNext;
head.setNext(current.getNext);
current.getNext.setPrev(head);
sz --;
}


}


public void forget(int k){
for(int i = 0; i<k; i++){
if(sz<=0){return;}


else{
Node<String> temp = dummy.getPrev;
dummy.setPrev(temp.getPrev);
}
}
}
```

```
public static void main(String [] args){
ForgetfullStack s = new ForgetfullStack();


s.push("A");
s.push("B");
s.push("C");
s.push("D");
s.push("E");
s.push("F");
s.pop();
s.forget(2);
}
}
```

Every method (besides forget) is taken from already confirmed O(1) linked list methods. The reason why ‘forget’ is O(1) is because it does not traverse the whole stack. All it does is repeat equal to the number given in main. Therefore, since said input is an integer, the time complexity simplifies to O(1).