3383 A1

1)a) Algorithm: isPalindrome(stringToCheck)
length = stringToCheck.length()
if(length == 0)
return "yes"
return isPalindromeRecursive(stringToCheck,
0, length-1)

Algorithm: isPalindrome Recursive(stringToCheck, startingIndex,
endingIndex)
if(startingIndex == endingIndex)
return "yes"
if((stringToCheck.charAt(startingIndex)) != (stringToCheck.charAt(endingIndex)))
return "No"
if(startingIndex < endingIndex+1)
return isPalindromeRecursive(stringToCheck
, startingIndex+1,
endingIndex -1)
return "yes"

2)c) iteration :- $1 \mid 2 \mid 3 \mid 4 \mid \dots \mid k^{th}$

j :- $2^1 \mid 2^2 \mid 2^3 \mid 2^4 \mid \dots \mid 2^k$

$$2^k = n$$
$$\log_2 2^k = \log_2 n$$
$$k = \log_2 n$$

$x = 2$	1
$j = 2$	1
while $(j \leq n)$	$\log n$
$x = x + 1$	$\log n$
$j = j \times 2$	$\log n$

$\Theta(\log n)$

2)d)

$x = 0$	1
$j = n$	1
while $(j >= 1)$	$\log n$
$x = x + 1$	$\log n$
$j = 2j/3$	$\log n$

$\Theta(\log n)$

iteration : $1 \mid 2 \mid 3 \mid 4 \mid \dots \mid k^{th}$

j : $n \mid (2/3)^1 n \mid (2/3)^2 n \mid (2/3)^3 n \mid \dots \mid (2/3)^{k-1} n = 1$

$$(2/3)^{k-1} n = 1$$
$$(2/3)^{k-1} = \frac{1}{n}$$
$$k - 1 = \log_{2/3} \frac{1}{n}$$
$$k = \log_{2/3} \frac{1}{n} + 1$$

$(n^{1/2})^{1/2} = n^{\frac{1}{2^2}}$

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

2f)

$x = 0$	1
$j = n$	1
while $(j \geq 2)$	loglogn
$x = x+1$	loglogn
$j = \sqrt{j}$	loglogn

$\Theta(\text{loglogn})$

iteration:	1	2	3	4	.....	k
j :	$n^{1/2^0}$	$n^{1/2^1}$	$n^{1/2^2}$	$n^{1/2^3}$	.....	$n^{1/2^{k-1}}$

$n^{1/2^{k-1}} = 2$

$\log_2 n^{1/2^{k-1}} = \log_2 2$

$1/2^{k-1} \log_2 n = 1$

$\log_2 n = 2^{k-1}$

$\log_2 \log_2 n = k-1 \log_2 2$ $\quad \log_a a = 1$

$\log_2 \log_2 n + 1 = k$

2a)

$p = 1$	1
for $i = 1$ to $5n^2$ do	n^2
$p = p \times i$	n^2
return p	1

$\Theta(n^2)$

iteration:	1	2	3	4	.....	k
:	1	2	3	4	.....	k

$k = 5n^2$

2e)

$x = 0$	1	
$j = 2$	1	
while $(j \leq n)$	$\log\log n$	$\Theta(\log\log n)$
$x = x + 1$	$\log\log n$	
$j = j^3$	$\log\log n$	

iteration:	1	2	3	4	.....	k^{th}
j :	2	2^3	2^{3^2}	2^{3^3}		$2^{3^{k-1}}$

$$2^{3^{k-1}} = n$$

$$3^{k-1} \log_2 2 = \log_2 n$$

$$3^{k-1} = \log_2 n$$

$$k - 1 = \log_3 \log_2 n$$

$$k = \log_3 \log_2 n + 1$$

2)g)

```
x = 0                          1
j = n                          1
while (j >= 1)                 n
    For i = j to n             n^2        Θ(n^2)
        x = x + 1              n^2
    j = j - 1                  n
return x                       1
```

iteration:	1	2	3	4	.....	k
j :	n	n-1	n-2	n-3		n-(k-1)

$n-(k-1) = 1$

$-(k-1) = 1-n$

$k-1 = -1+n$

$k = n$

~~for loop~~ :

j:	n	n-1	n-2	n-3	.....	n-(k-1)
For loop:	0	1	2	3		k-1

$k-1 = n$

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

2b)

```
s = 0                          1
For i = 1 to n                 n
    For j = i to n             n^2
        s = s + 1              n^2
return s                       1
```

$\Theta(n^2)$

iter:	1	2	3	4	...	k
i :	1	2	3	4		k
j :	1	2	3	4		k

$k = n$

$$\text{innerloop} = \frac{n(n+1)}{2} = n^2$$

4 a & c) Algorithm: repeatNums (array)

```
n = array.length                    1
for (i ← 0; i < n    )              n
    m = array[i] % n                n
    array[m] += n                   n
For (j ← 0; j < n    )              n      Θ(n)
    if (array[j] ≥ 2*n)             n
        print(j)                    n
```

iteration:	1	2	3	4	...	k
i :	0	1	2	3		n-1

$k = n-1$

iteration:	1	2	3	4	...	k
j :	0	1	2	3		n-1

$k = n-1$

3a)

```
s ← 0                                   1
for i ← 0 to n-1                        n
    for j ← i to n-1                    n
        p ← 1                           n        Θ(n^3)
        for k ← i to j                  n
            p ← p * A[k]                n
        s ← s + p                       n
return s                                1
```

iteration:	1	2	3	4	.....	k
i :	0	1	2	3		k-1

k = t = n → t = n

j :	0	0	1	2	k
2nd loop :					

i :	0	1
j :	0	1
3rd loop :	0	

2nd loop : i :	0	1	2	.....	k
	n-1	n-2	n-3		0

i, j, k are incremented by a constant