GTU Department of Computer Engineering

CSE 222 / 505 – Spring 2022

Homework 2

Ahmet USLUOGLU

1801042602

1) For each of the following statements, specify whether it is true or not, and prove your claim. Use the definition of asymptotic notations.

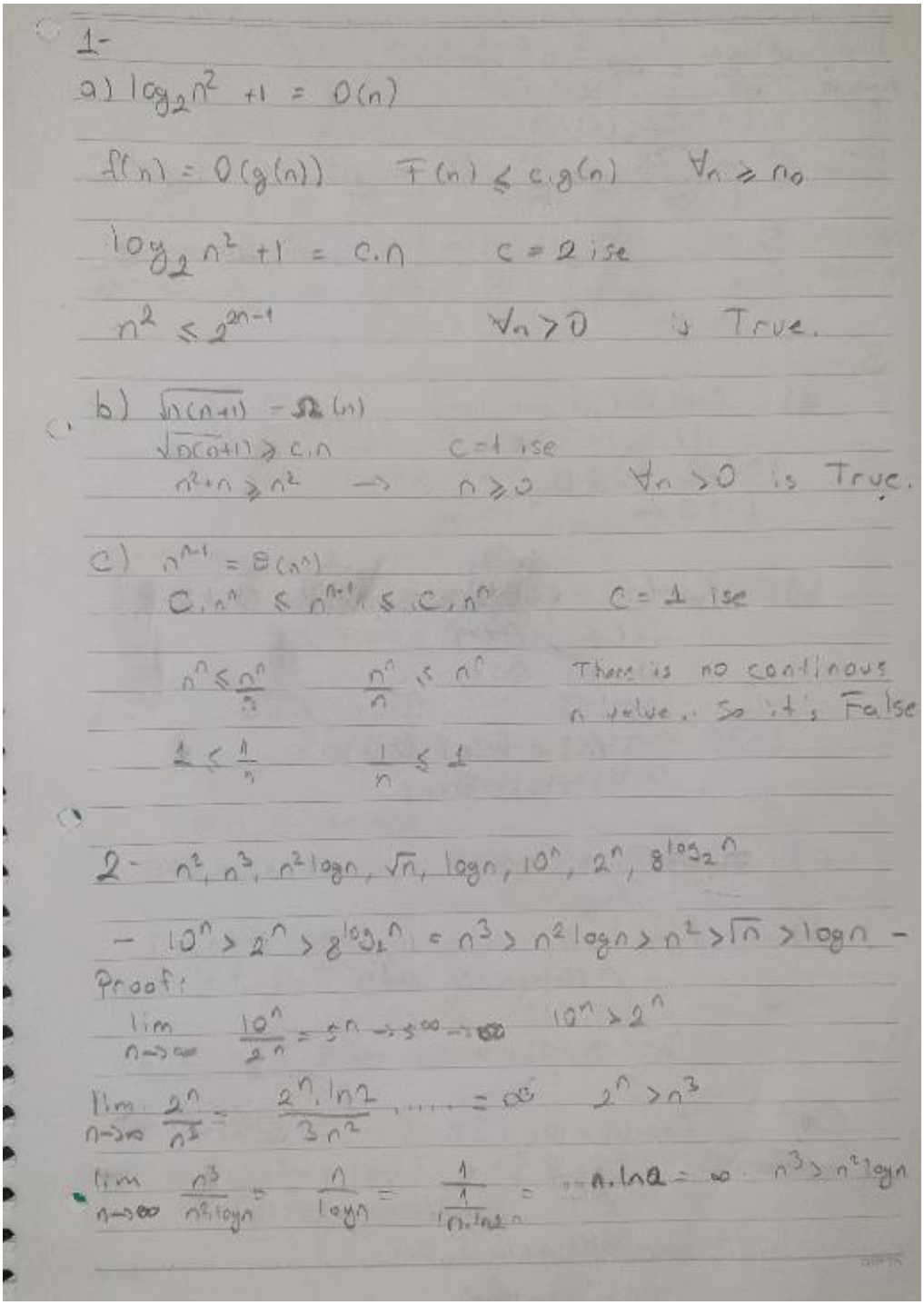
a) $\log_2 n^2 + 1 = O(n)$

b) $\sqrt{n(n+1)} = \Omega(n)$

c) $n^{n-1} = \theta(n^n)$

2) Order the following functions by growth rate and explain your reasoning for each of them. Use the limit method.

$n^2,\ n^3,\ n^2 \log n,\ \sqrt{n},\ \log n,\ 10^n,\ 2^n,\ 8^{\log_2 n}$



3) What is the time complexity of the following programs? Use most appropriate asymptotic notation. Explain by giving details.

$\lim_{n\to\infty} \frac{n^2 \log n}{n^2} = \infty \quad n^2 \log n > n^2$

$\lim_{n\to\infty} \frac{n^2}{\sqrt{n}} \to \frac{2n}{\frac{1}{2} n^{-\frac{1}{2}}} = \frac{n}{\frac{1}{\sqrt{n}}} = n\sqrt{n} = n^2 > \sqrt{n}$

$\lim_{n\to\infty} \frac{\sqrt{n}}{\log n} = \frac{\frac{1}{2\sqrt{n}}}{\frac{1}{n \cdot \ln 2}} = \frac{\sqrt{n} \cdot \ln 2}{2} = \infty \quad \sqrt{n} > \log n$

3.

a)
```
for(i=2, i<=n, i++)
    if(i%2==0) c++;
    else i=(i-1)^2;
```

b)
```
for(i=0, i<n, i++)   Θ(n)
    if()   Θ(1)
    else()   Θ(1)
```

$T(n) = \Theta(n) \cdot \Theta(1)$

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

c) `return;` $\Theta(1)$ $\quad T(n) = \Theta(1)$

d)
```
for(i=0, i<n, i=i+5)   Θ(n/5)
    /* statement */   Θ(1)
```

$T_{on} = \Theta(n/5) \cdot \Theta(1) \to \underline{T(n) = \Theta(n)}$

e)
```
for(i=0, i<n, i++)   Θ(n)
    for(j=1, j<i; j=j*2)   Θ(log n)
        printf();   Θ(1)
```

$T(n) = \Theta(n) \cdot \Theta(\log n) \cdot \Theta(1)$

$\underline{T(n) = \Theta(n \log n)}$

f) if(p_4( ) > 1000) $\Theta(n)$
p_5( ) $\Theta(n \log n)$
else printf(P_3(), P_4()); $\Theta(n) + \Theta(1)$

worst case → $T_1(n) = \Theta(n) + \Theta(n \log n)$
$T_1(n) = \Theta(n \log n)$ → $O(n \log n)$

Best case → $T_2(n) = \Theta(n) + \Theta(n)$
$T_2(n) = \Theta(n)$ → $\Omega(n)$

g) i = n;
while(i > 0) $\Theta(\log n)$
for(j = 0, j < n, j++) $\Theta(n)$
print( ); → $\Theta(1)$

$\Theta(1) \leftarrow i = i/2$

$T(n) = \Theta(\log n) * \Theta(n) . \Theta(1)$
$T(n) = \Theta(n \log n)$

h) while (n > 0) → $\Theta(\log n)$
for (j = 0, j < n, j++) → $\Theta(n)$
print( ) → $\Theta(1)$

$\Theta(1) \leftarrow n = n/2$

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

i) int p_9(n)
if (n = 0) return 1; → $\Theta(1)$
else return n * p_9(n-1) → $\Theta(n-1)$

n = 0 1
n > 0 $T(n-1) + 1$
$T(n) = T(n-k) + k$

n = k
$T(n) = \underbrace{T(0)}_{1} + n$
$T(n) = 1 + n$
$T(n) = \Theta(n)$

4)

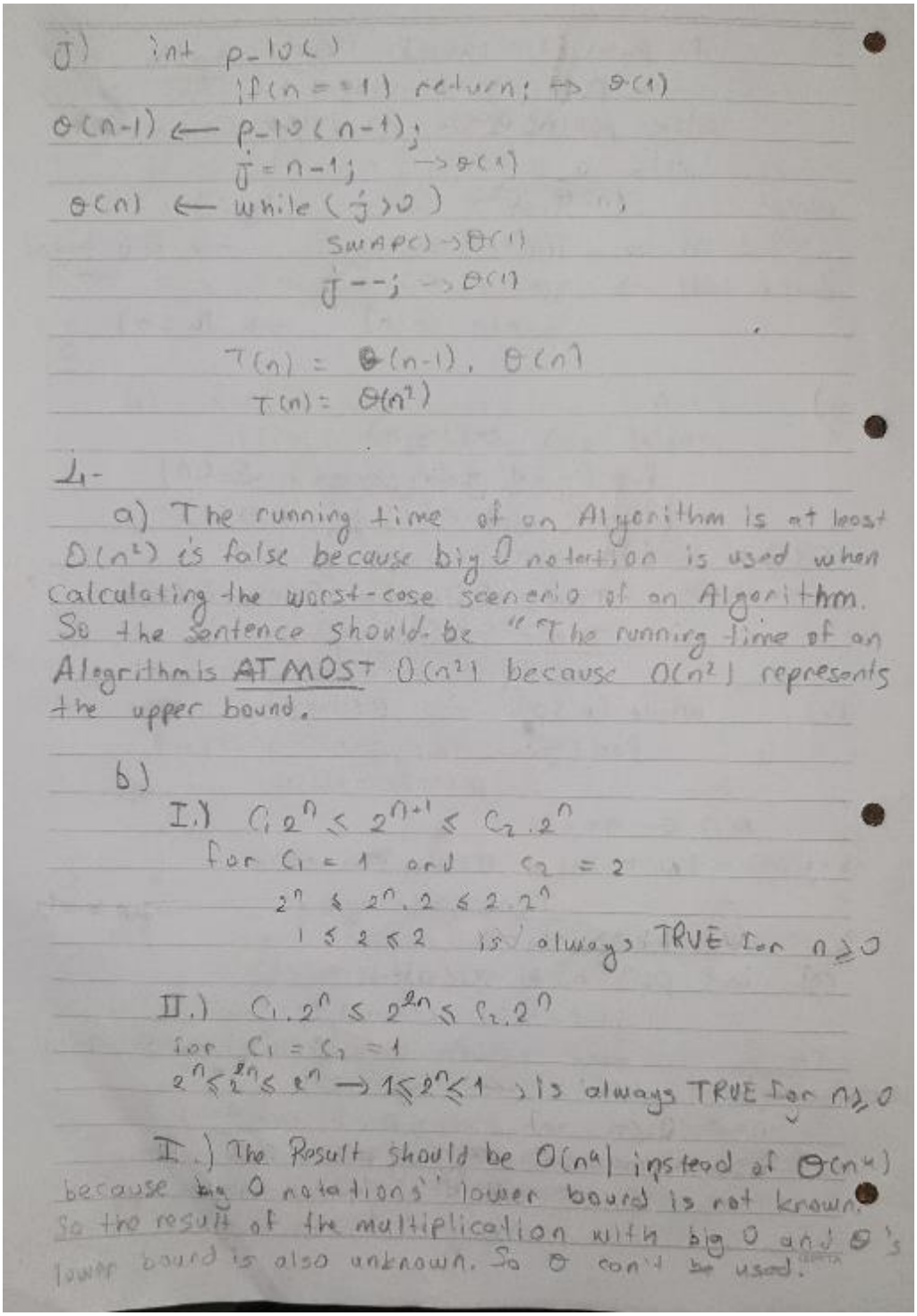
a) Explain what is wrong with the following statement. "The running time of algorithm A is at least $O(n^2)$".

b) Prove that clause true or false? Use the definition of asymptotic notations.

I. $2^{n+1} = \Theta(2n)$

II. $2^{2n} = \Theta(2n)$

III. Let $f(n)=O(n^2)$ and $g(n)= \Theta(n^2)$. Prove or disprove that: $f(n) * g(n) = \Theta(n^4)$



3)

```
int p_10(j)
    if(n==1) return; → Θ(1)
Θ(n-1) ← p_10(n-1);
    j = n-1; → Θ(1)
Θ(n) ← while(j>0)
        SWAP() → Θ(1)
        j--; → Θ(1)
```

$T(n) = \Theta(n-1) \cdot \Theta(n)$

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

4-

a) The running time of an Algorithm is at least $O(n^2)$ is false because big O notation is used when calculating the worst-case scenario of an Algorithm. So the sentence should be "The running time of an Algorithm is ATMOST $O(n^2)$ because $O(n^2)$ represents the upper bound.

b)

I.) $C_1 2^n \leq 2^{n+1} \leq C_2 \cdot 2^n$

for $C_1 = 1$ and $C_2 = 2$

$2^n \leq 2^n \cdot 2 \leq 2 \cdot 2^n$

$1 \leq 2 \leq 2$ is always TRUE for $n \geq 0$

II.) $C_1 \cdot 2^n \leq 2^{2n} \leq C_2 \cdot 2^n$

for $C_1 = C_2 = 1$

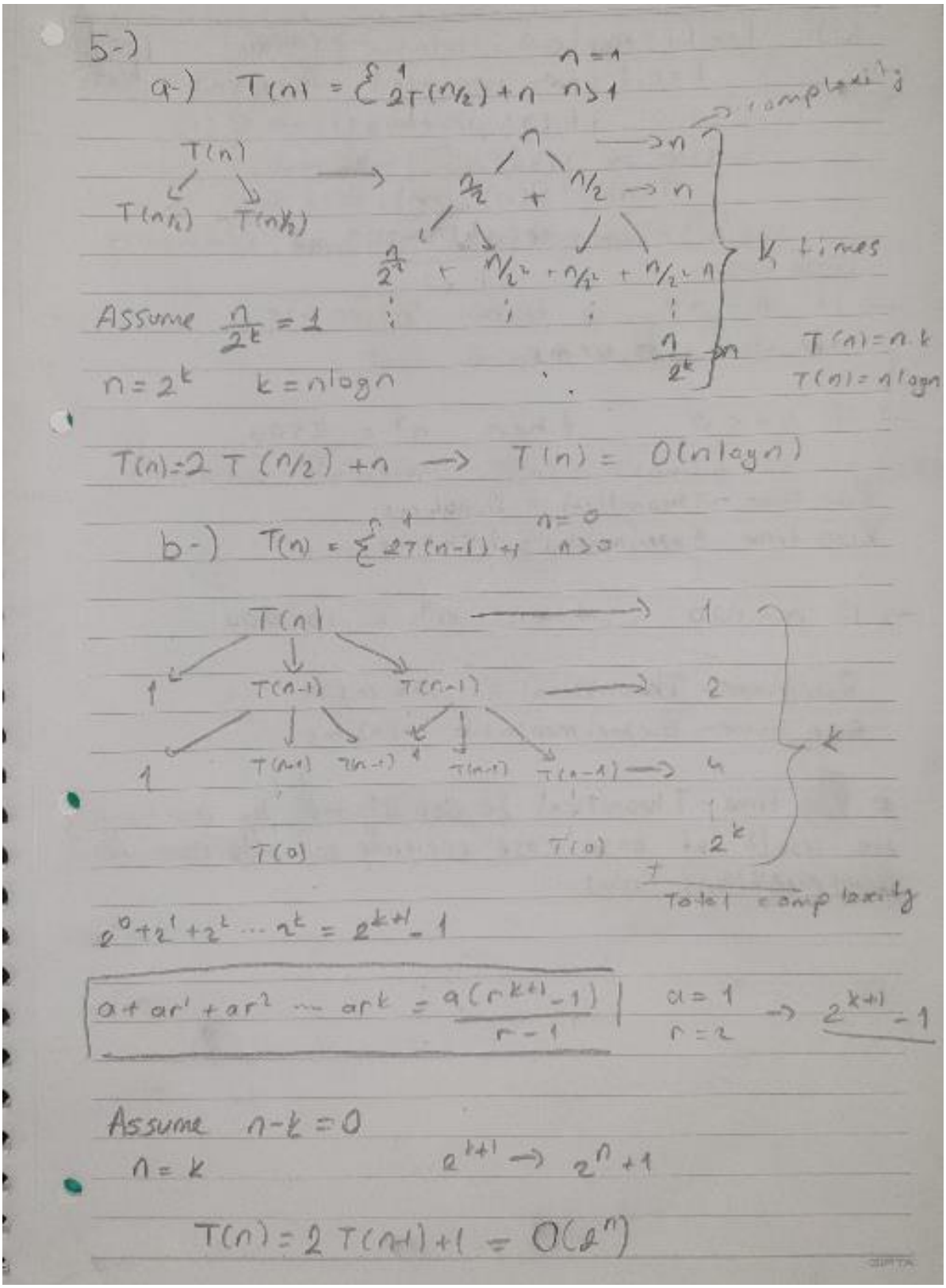
$2^n \leq 2^{2n} \leq 2^n \rightarrow 1 \leq 2^n \leq 1$ is always TRUE for $n \geq 0$

III.) The Result should be $O(n^4)$ instead of $\Theta(n^4)$ because the O notations' lower bound is not known. So the result of the multiplication with big O and Θ's lower bound is also unknown. So Θ can't be used.

5) Solve the following recurrence relations. Express the result in most appropriate asymptotic notation. Show details of your work.

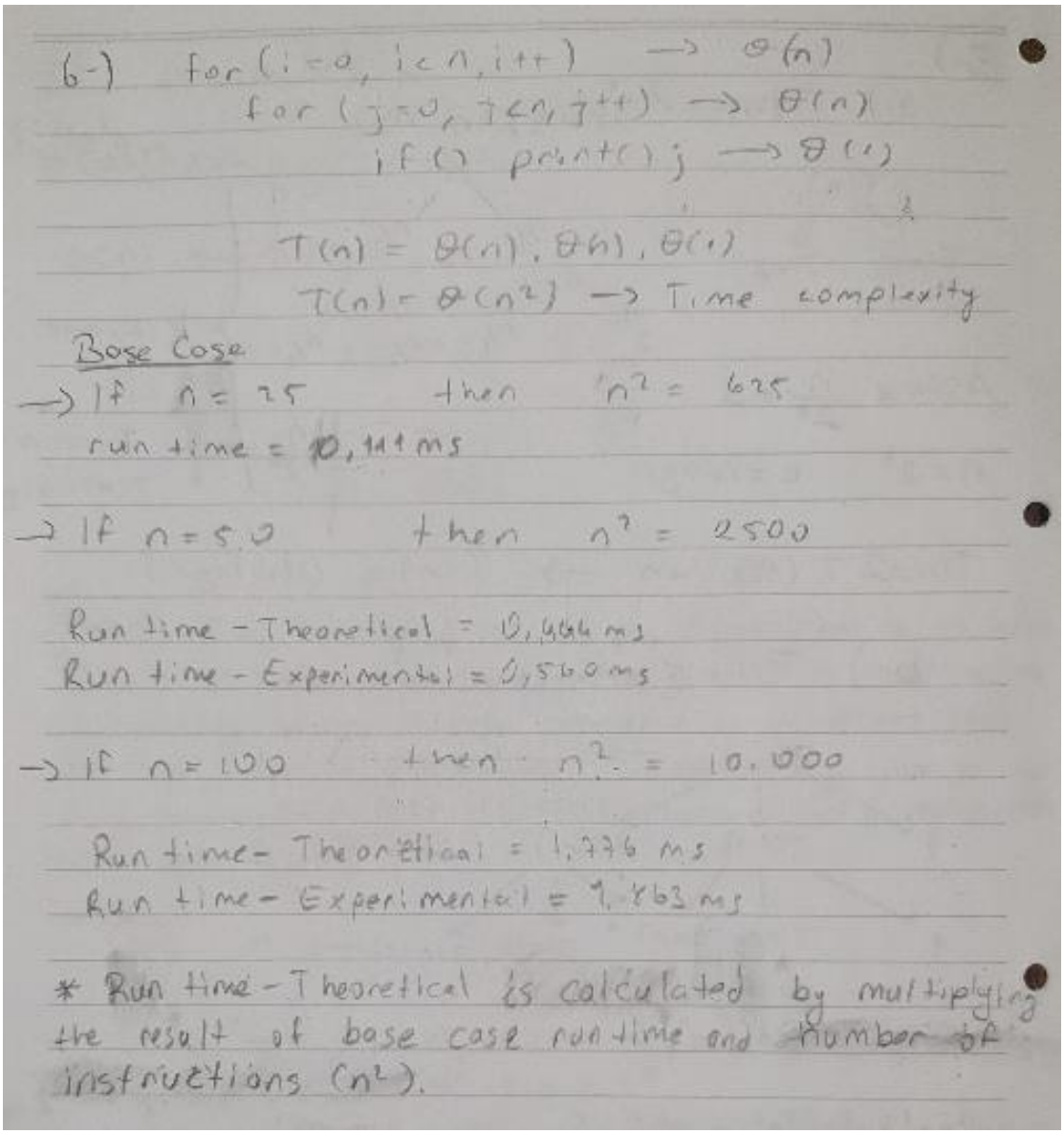
a) T(n) = 2T(n/2) + n, T(1) = 1

b) T(n) = 2T(n − 1) + 1, T(0)=0



6) In an array of numbers (positive or negative), find pairs of numbers with the given sum. Design an iterative algorithm for the problem. Test the algorithm with different size arrays and record the running time. Calculate the resulting time complexity. Compare and interpret the test result with your theoretical result.

```java
/**
 * Given an array of integers, find the number of pairs of integers in the array that sum to a
 * given total
 *
 * @param arr the array to be searched
 * @param total the sum of the two integers you're looking for
 */
static void Sum_Iteration(int[] arr, int total)
{
    int count = 0;
    for(int i = 0; i < arr.length; i++)
        for(int j = 0; j < arr.length; j++)
            if(arr[i] + arr[j] == total)
                count++;
}
```



```
ahmet@AUslu-Legion:/mnt/c/Users/us7/Desktop/hw2 data$ javac hw2.java
ahmet@AUslu-Legion:/mnt/c/Users/us7/Desktop/hw2 data$ java hw2
Iteration

Array with n = 25 elements :0.111ms
Array with n = 50 elements :0.599ms
Array with n = 100 elements :2.083ms

Recursion

Array with n = 25 elements :0.915ms
Array with n = 50 elements :3.096ms
Array with n = 100 elements :5.612ms
ahmet@AUslu-Legion:/mnt/c/Users/us7/Desktop/hw2 data$ java hw2
Iteration

Array with n = 25 elements :0.115ms
Array with n = 50 elements :0.389ms
Array with n = 100 elements :3.057ms

Recursion

Array with n = 25 elements :1.425ms
Array with n = 50 elements :3.368ms
Array with n = 100 elements :7.105ms
ahmet@AUslu-Legion:/mnt/c/Users/us7/Desktop/hw2 data$ java hw2
Iteration

Array with n = 25 elements :0.106ms
Array with n = 50 elements :0.881ms
Array with n = 100 elements :2.354ms

Recursion

Array with n = 25 elements :0.379ms
Array with n = 50 elements :3.124ms
Array with n = 100 elements :5.87ms
ahmet@AUslu-Legion:/mnt/c/Users/us7/Desktop/hw2 data$ java hw2
Iteration

Array with n = 25 elements :0.111ms
Array with n = 50 elements :0.56ms
Array with n = 100 elements :1.863ms

Recursion

Array with n = 25 elements :0.683ms
Array with n = 50 elements :2.6ms
Array with n = 100 elements :6.597ms
ahmet@AUslu-Legion:/mnt/c/Users/us7/Desktop/hw2 data$
```

7) Write a recursive algorithm for the problem in 6 and calculate its time complexity. Write a recurrence relation and solve it.

```java
/**
 * Given an array of integers, find the number of pairs of integers in the array that sum to a
 * given value
 *
 * @param arr The array of integers.
 * @param total The sum we're looking for
 * @param index1 The index of the first element in the array.
 * @param index2 The index of the second element in the array.
 */
static void Sum_Recursive(int[] arr, int total, int index1, int index2)
{
    int count = 0;
    if (index1 == arr.length) return;

    if(index2 < arr.length)
    {
        if(arr[index1] + arr[index2] == total) count++;

        Sum_Recursive(arr, total, index1, index2 + 1);
    }
    else
    {
        index1++;
        index2 = index1;
        Sum_Recursive(arr, total, index1, index2);
    }
}
```