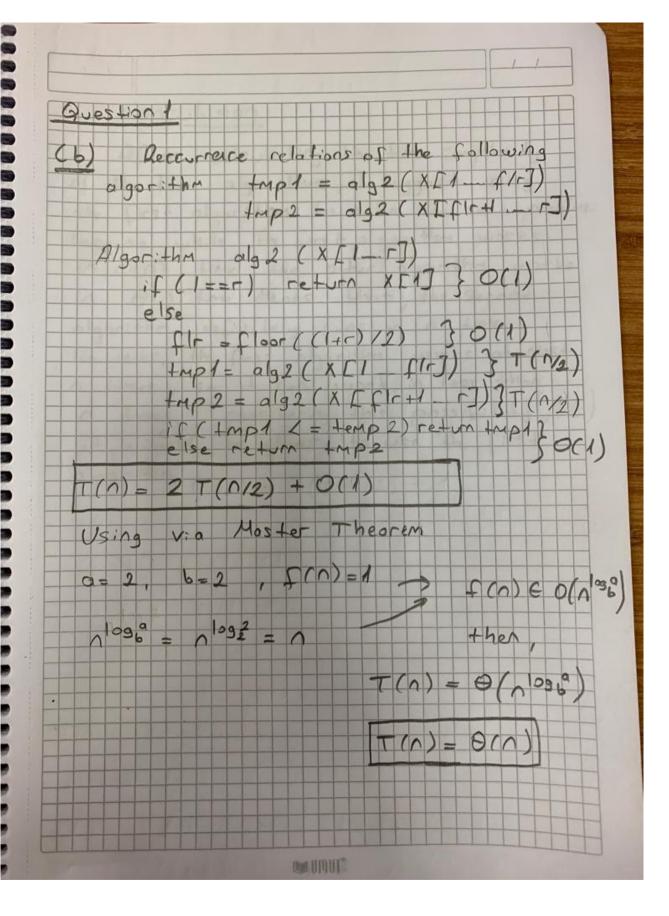
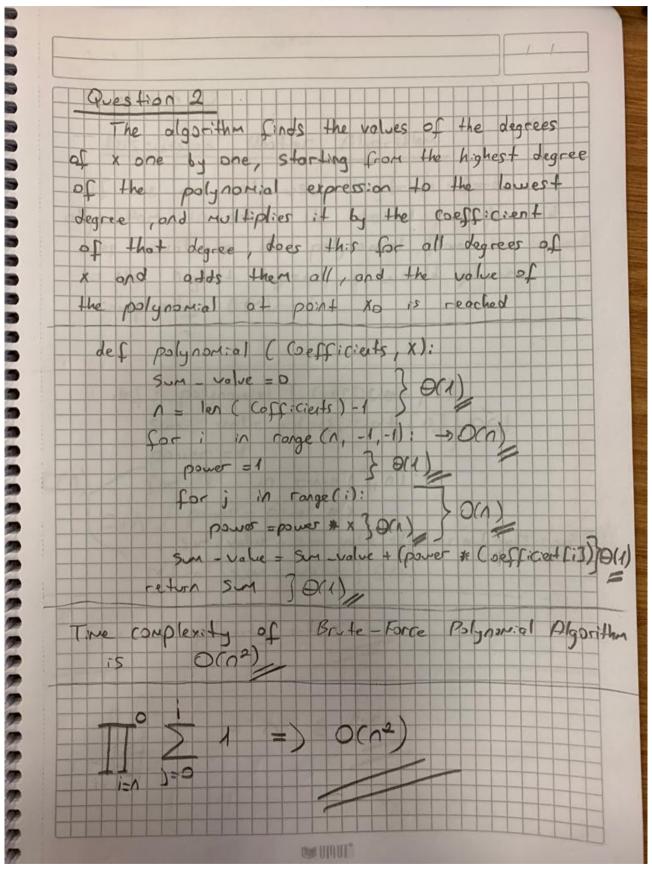
PART 1

None / Surname & Yunus Emre Gegik	ME
No 3 1801042635	
CSE 321 - HW3	
Question 1	
9025+1011	
(a) Recurrence relations of the Collowing	algor: thm
is +mp = alg1 (L[0 n-2])	
Algorithm alg1 (110-0-17)	
if (n = = 1) return 150] } 00	1
else 1100000000000000000000000000000000000	1-1)
7MP = 0191 (220-112)	
	+MP (0(1)
else neturn AIn-11	
T(n) = T(n-1) + O(1)	then n=1
	(1)=1
(n) = 1 (n + 1) + 1 + 1 + 1 + 1	
T(A) (T(A-2) +1) +1	T(0-2)+1
1(0-2)	7(1-3)+1
T(n) = (T(n-3)+1) + 2	
= T(n-3) + 3	
i k 1 mes	
	-11001
T(n) = T(n-k) +k	(v) E (v)
Assume that n= k+1	
r(n) = T(n-(n-1)) + (n-1)	
T(n) = T(1) + (n-1) = T(n)	= 1

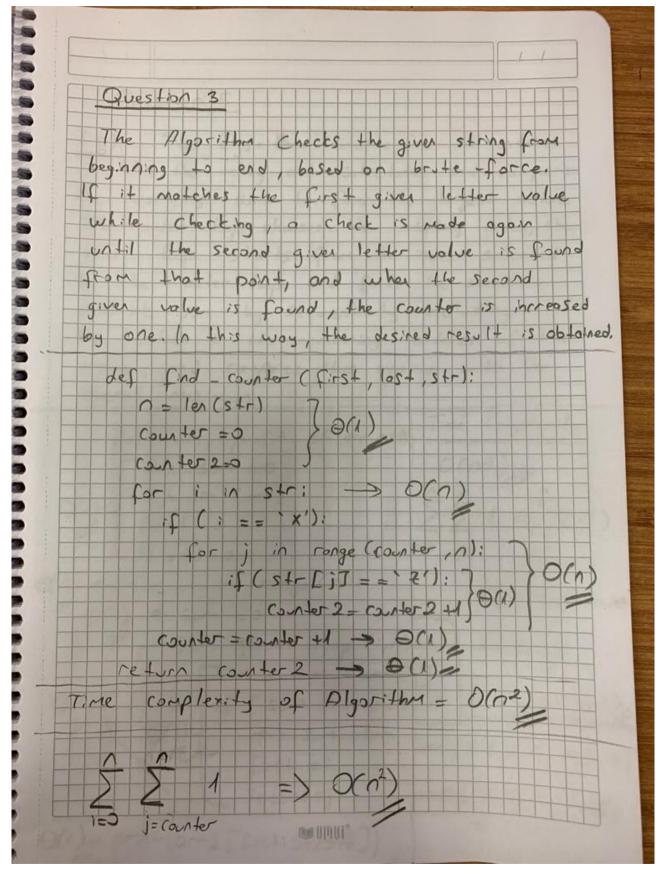


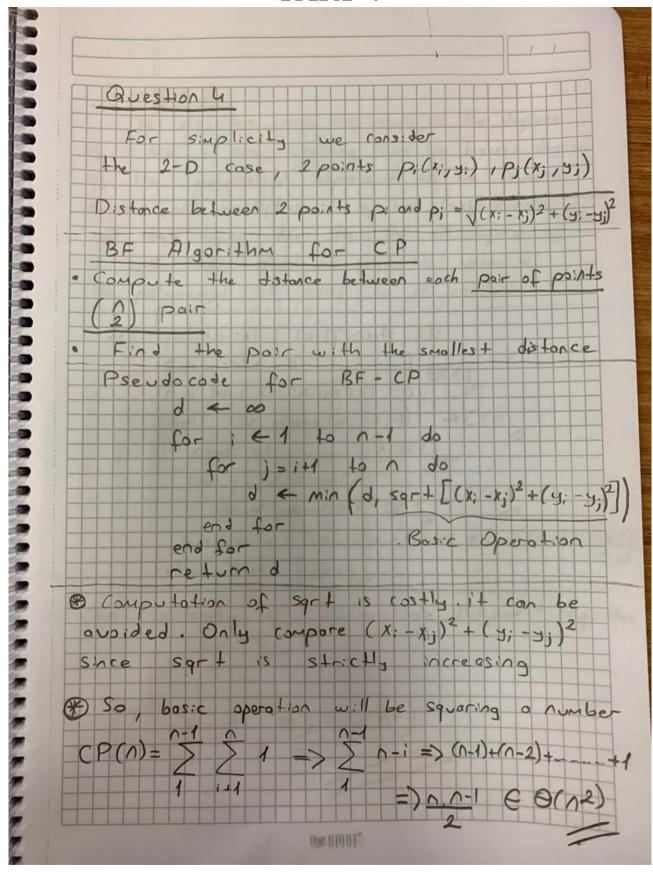
Conculusion After colculating the time complexity of the two algorithms, I suggest using the first algorithm to solve the some problem. Because while the first olgorithm works in O(n) time, the other olgorithm works in O(n) time. Means that, the second algorithm definitely runs in a time. However, the first algorithm works at most n times, so it can work in less than n times, considering this possibility, it would be cliver to use the first algorithm.

## PART 2



Question -2 Discussion Another Algorithm If we did not use the brute-force algorithm to solve the desired problem and does evaluate the polynomial from the lowest to the highest degree, we would get time complexity ocal. def polynomal ( Gefficient, x) Sum value = Coefficient [0] O(1) power =1 orn) for i in range (1, n). power = power \* x Sur value = sur value + (power \* (seff: cient [i]) return sur- value } arx) Complexity = O(n) e orn) DE UMUT"





## PART 5

