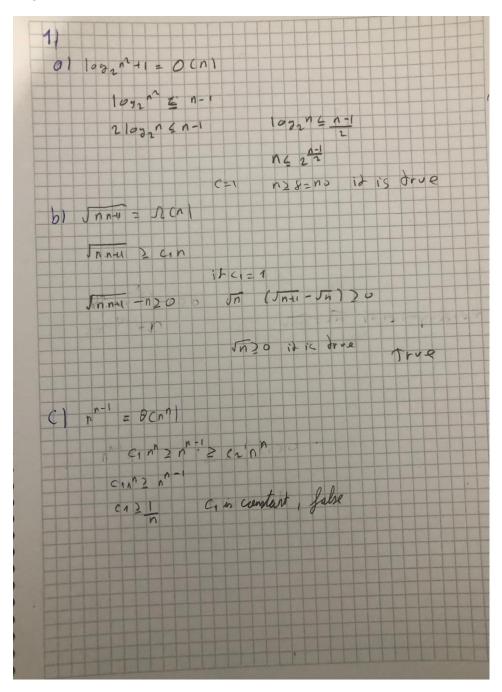
MUHAMMET ÇAĞRI YILMAZ 1901042694

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

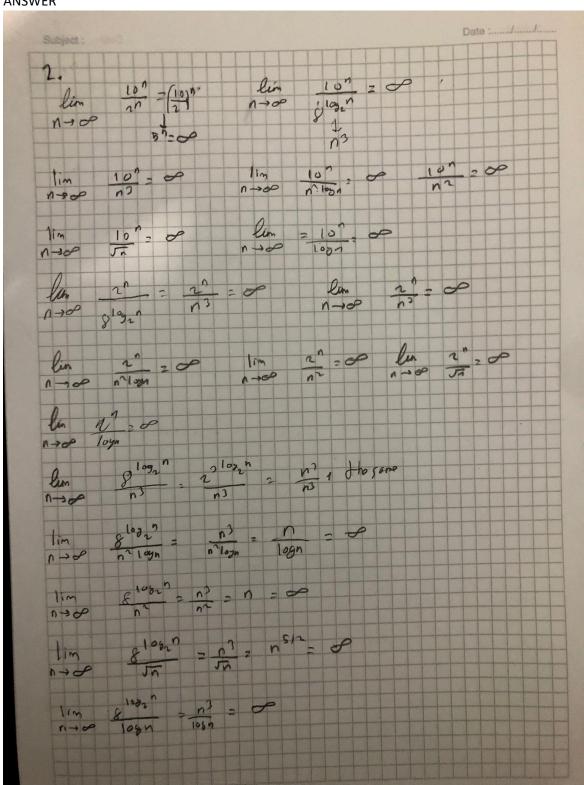
b)
$$\sqrt{n(n+1)} = \Omega(n)$$

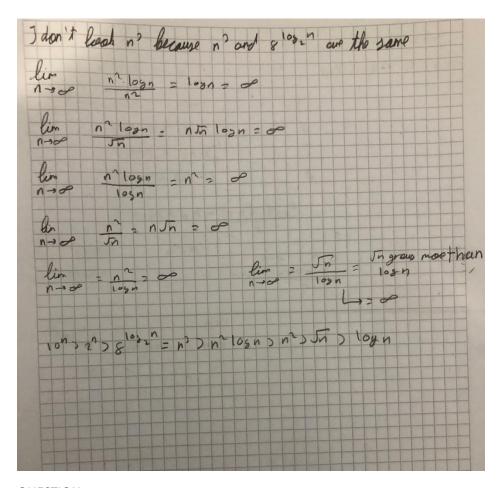
c)
$$n^{n-1} = \theta(n^n)$$



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 , $8^{\log_2 n}$



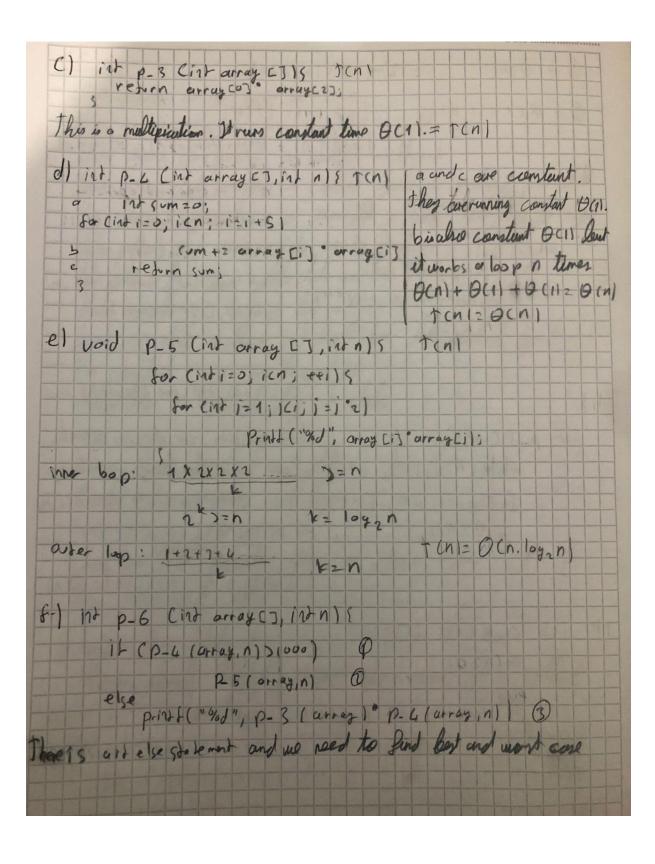


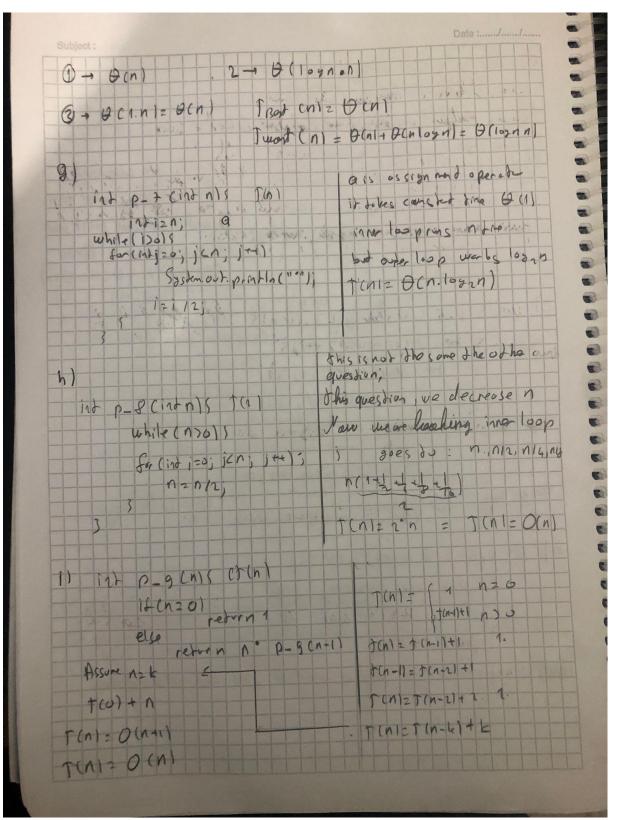
3) What is the time complexity of the following programs? Use most appropriate asymptotic notation. Explain by giving details.

```
a)
int p_1 ( int my_array[]){
                                                        for(int i=2; i<=n; i++){
                                                                                                         if(i%2==0){
                                                                                                                                                         count++;
                                                                                                                                                              i=(i-1)i;
   }
b)
   int p_2 (int my_array[]){
                                                   first_element = my_array[0];
                                                   second_element = my_array[0];
                                                      for(int i=0; i<sizeofArray; i++){
                                                                                                        if(my_array[i]<first_element){
                                                                                                                                                           second_element=first_element;
                                                                                                                                                           first_element=my_array[i];
                                                                                                        \label{lem:lement} \label{lem:lement} \mbox{$\ensuremath{\text{element}}$} \mbox{$\ensuremath{\text{eleme
                                                                                                                                                         if(my_array[i]!= first_element){
                                                                                                                                                                                                                second_element= my_array[i];
```

```
c)
int p_3 (int array[]) {
        return array[0] * array[2];
d)
int p_4(int array[], int n) {
        Int sum = 0
        for (int i = 0; i < n; i=i+5)
                sum += array[i] * array[i];
        return sum;
}
e)
void p_5 (int array[], int n){
        for (int i = 0; i < n; i++)
                for (int j = 1; j < i; j=j*2)
                       printf("%d", array[i] * array[j]);
}
f)
int p_6(int array[], int n) {
        If (p_4(array, n)) > 1000)
               p_5(array, n)
        else printf("%d", p_3(array) * p_4(array, n))
 g)
 int p_7( int n ){
           int i = n;
           while (i > 0) {
                    for (int j = 0; j < n; j++)
                              System.out.println("*");
                     i = i / 2;
            }
 }
 h)
 int p_8( int n ){
           while (n > 0) {
                    for (int j = 0; j < n; j++)
                               System.out.println("*");
                     n = n / 2;
            }
 }
```

```
3-1
a) int p 1 (int my array () () There is a if else statement. In if
                                        estatement and else statement . in else
           for cintizz; ic=n; i+1) 5
                                        estatement. i does not increuse
               14(1%2==0)5
                                        linearly, it increuses 12. there-
                  count ++
            3 elses : (1-1) i;
                                         Store, O(10 2 2 n)
      ind p-2 (int my-array co) ( T(n)
      first elevent z my array (0);
     second-elements my_orray (0);
      for (nhi=u; 1 ( size of Armoy; +ti)s
             id (my orreg [i] Ldirst elevent)
                       secund elements first element;
             selse is company cists fort-elevatis
                           Secund-element my orranciz;
   All statements that we in loop take constant two $\text{O(1)} . Socy runs ntine other statements run coenstant two or well. $\text{O(1)} + \text{O(1)} + \text{O(1)} = \text{O(n)}=
                        + (n12 Ocm
```





ind p-10 Cint ACT, ind nis T(n) 12 (n==1) redurn; P-10 (A, n-1); while (j) o and ACj] (ACj-1)) SWAP (AC)], ACj-13); n= 1 1(1)2 T (n-1)+n+1, 1 T(n)= T(n-1)+n+1 1. step Ten-11= 1 (n-2)+n T(n)= T(n-2)+2n+1 2. step r(n-2)= T(n-3)+n-1 3. step 1 cm = 1 cn-21+30 T(n-3)= T(n-4)-1n-2 4. Step T(n)2 T(n-4)+4n-2 Ton 12 Ton-61+ kin + (1-(62)+1) 1/2 1) Sep Set assume +1-le=1 k=0-1 T(1)+ N2-1 -12+11-1 +(11+20=2n -n2+n-1 = n2-n-1 +1
2 T(n)= D(n2

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

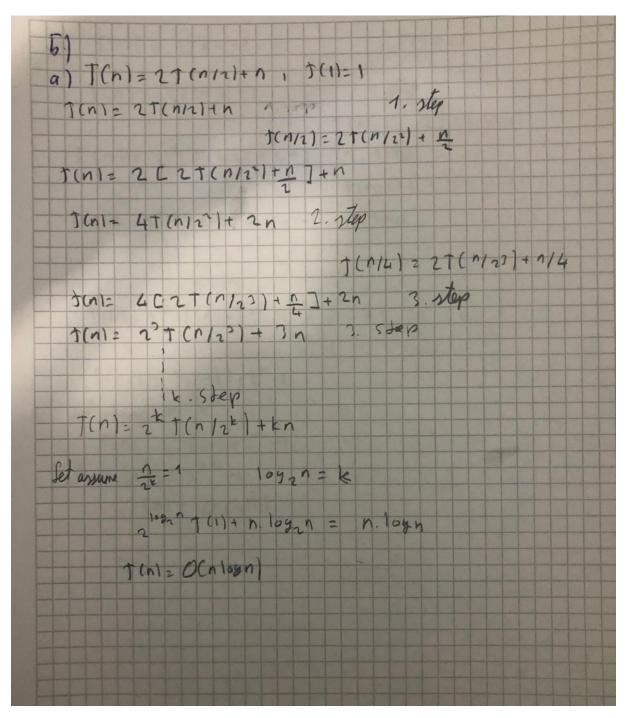
III. Let $f(n)=O(n^2)$ and $g(n)=\Theta(n^2)$. Prove or disprove that: $f(n)*g(n)=\Theta(n^4)$

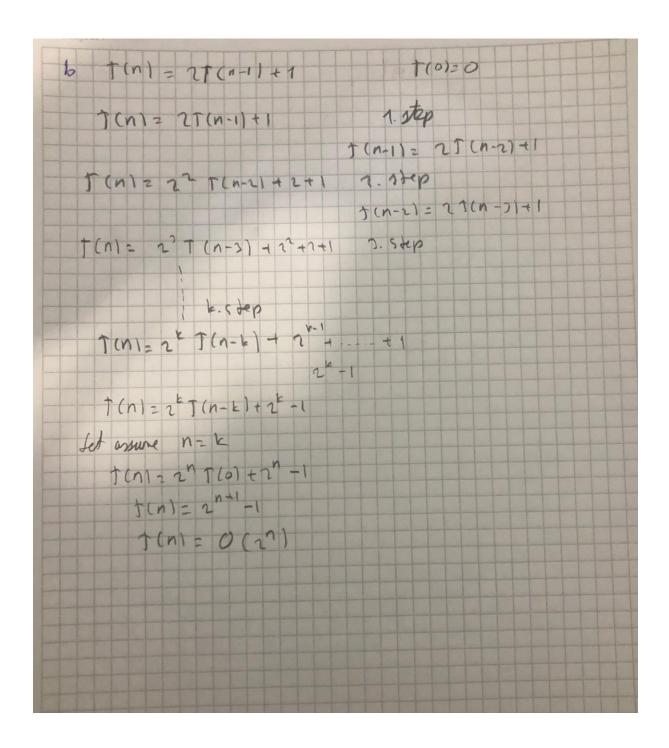
| an "upper bound" for the function. If this statemen "Therunning time of algorithm A wat max OCM", it is be true. | mercieles of mass would |
|--|-------------------------------|
| 6-1 | |
| 1 2n+1 = 0 (2 n) | |
| 2 ⁿ⁺¹ = 0(2 ⁿ) (m/C 1 2 ⁿ) = 2 ⁿ = 2 (2 2 ⁿ), C ₁ , C | (2)0 |
| Now I assign 1 to n n=1 | DIF. |
| C122.4 42 C2.2 | |
| 21 0 (122 22 22 2 | 4400 |
| 2. 2 ⁿ ₂ B(2n) | |
| | |
| e, 2 2 1 false since c1 is constant a s | |
| The sale | se |
| 3. SCN = Och means that & Cn) & cn + | 4 5 |
| 3. SCn)= Oca means that & Cn) & cn = 3 cn = 6 cm means that there is a strict when we there come a strict when we | multiply |
| mem (fin) (gin) was cordinary so this is Buy a | h cn4) |
| not teto false | |
| | |
| | |
| | |

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-)

QUESTION

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.

ANSWER

Assume that n is size of array.

If statement runs constant time because this is only a increment operation

Inner loop runs n times $\Theta(n*1) = \Theta(n)$;

Outter loop runs n times $\Theta(n^*n) = \Theta(n^2)$.

You can see code below.

Theoretically, Time complexity is $\Theta(n^2)$.

```
int[] arr2;
arr2 = new int[10000];
arr2[0]=10;
arr2[1]=3;
arr2[2]=5;
arr2[3]=2;
arr2[4]=4;
arr2[5]=-3;
int i;
for(i=6;i< arr2.length;++i){
    arr2[i]=i+10;
}
long startTime = System.currentTimeMillis();
System.out.println(checkPairs(arr2, |value: 7));
long endTime = System.currentTimeMillis();
long totalTime = endTime - startTime;
System.out.println("Running time "+totalTime);</pre>
```

I have 2 test cases one of them is that: Array size is 10000 the other one is that: Array size is 100000

When I run this program it gives 34 ms.

```
int[] arr2 ;
arr2 = new int[100000] ;
arr2[0]=10;
arr2[1]=3;
arr2[2]=5;
arr2[3]=2;
arr2[4]=4;
arr2[5]=-3;
int i;
for(i=6;i< arr2.length;++i){
    arr2[i]=i+10;
}
long startTime = System.currentTimeMillis();
System.out.println(checkPairs(arr2, value: 7));
long endTime = System.currentTimeMillis();
System.out.println("Running time "+totalTime);</pre>
```

However, It gives 4271 ms.

If I look theoretically, one of them 34 ms the other one will be 3400 ms.

But there are a lot of conditions that effect running time directly(compiler,OS).

QUESTION

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

Ton = T(n-1)+n , n>0 1. step T(n) = T(n-1)+n 1 (n-112 T (n-2) + n-1 7'(n) = T(n-2)+2n-1 2.5dep T(n-2) = T(n-3) + 1 n-2 T(n-3) = T(n-4) + 1 n-3 1cn121(n-2)+31-3 7.5ter 7 (n1- T(n-4) + 4n-6 4.5 Jep T(n = T(n = 1 + k n - 1 k k - 1 letopare n=k 1 (0) + n - 12-12 Ten = n - n - n + 1 tin = O(n2)