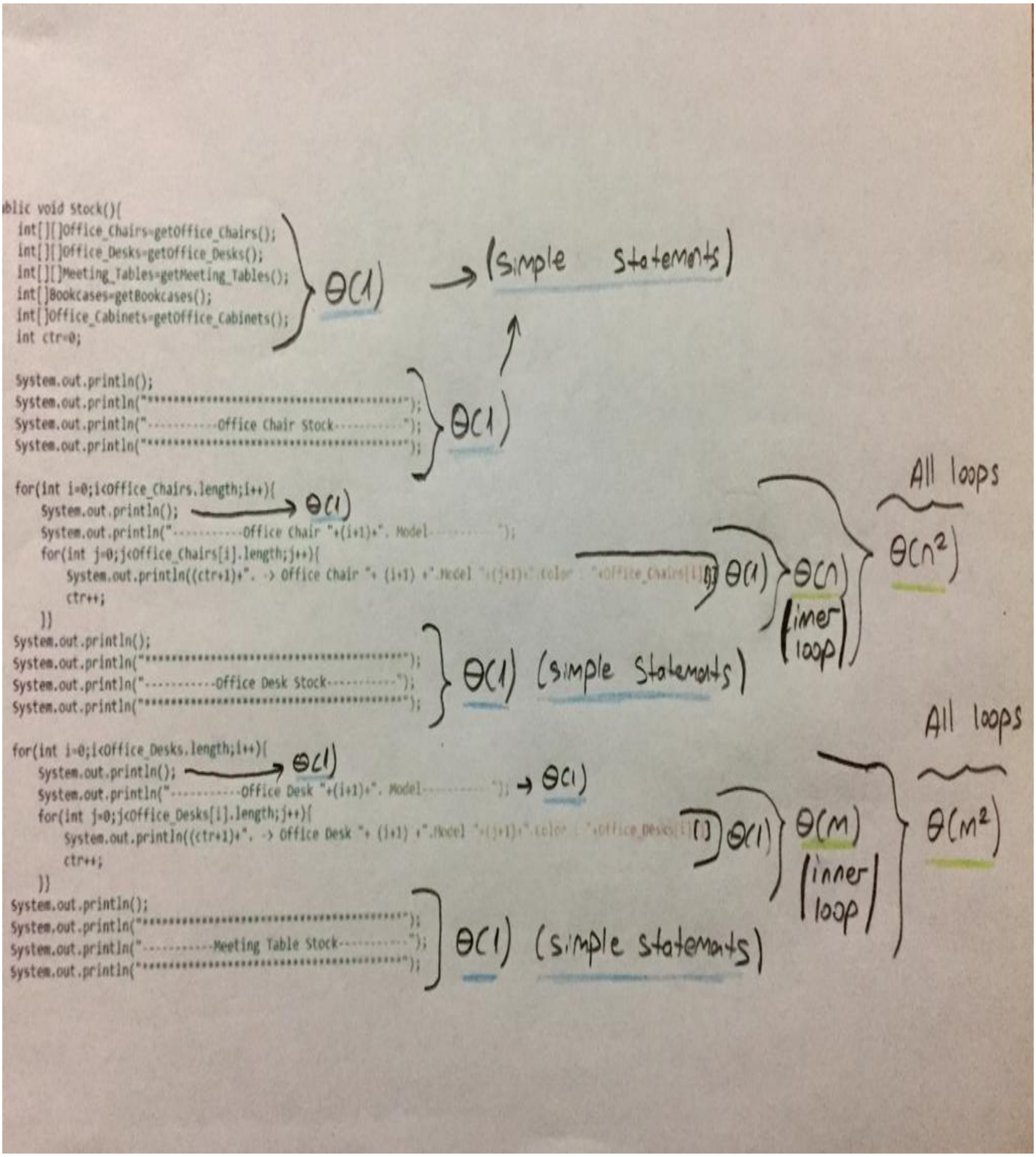
PART 1

1)Customer Search The Product In Stock



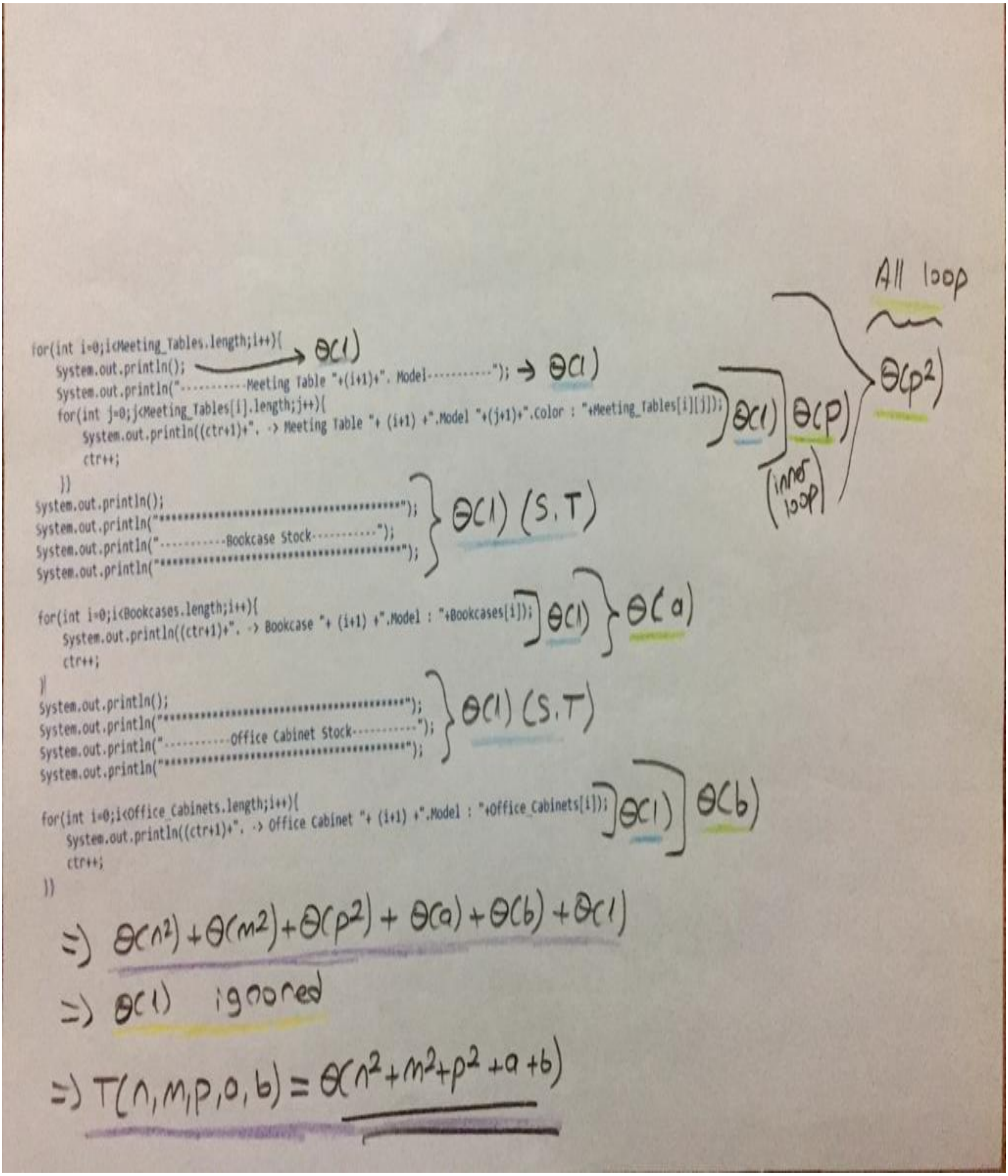
```
blic void Stock(){
  int[][]Office_Chairs=getOffice_Chairs();
  int[][]Office_Desks=getOffice_Desks();
  int[][]Meeting_Tables=getMeeting_Tables();
  int[]Bookcases=getBookcases();
  int[]Office_Cabinets=getOffice_Cabinets();
  int ctr=0;

  System.out.println();
  System.out.println("****************************************");
  System.out.println("-----------Office Chair Stock-----------");
  System.out.println("****************************************");

  for(int i=0;i<Office_Chairs.length;i++){
      System.out.println();
      System.out.println("-----------Office Chair "+(i+1)+". Model-----------");
      for(int j=0;j<Office_Chairs[i].length;j++){
          System.out.println((ctr+1)+". -> Office Chair "+ (i+1) +".Model "+(j+1)+".Color : "+Office_Chairs[i]
          ctr++;
    }}
  System.out.println();
  System.out.println("****************************************");
  System.out.println("-----------Office Desk Stock-----------");
  System.out.println("****************************************");

  for(int i=0;i<Office_Desks.length;i++){
      System.out.println();
      System.out.println("-----------Office Desk "+(i+1)+". Model-----------");
      for(int j=0;j<Office_Desks[i].length;j++){
          System.out.println((ctr+1)+". -> Office Desk "+ (i+1) +".Model "+(j+1)+".Color : "+Office_Desks[i][]
          ctr++;
    }}
  System.out.println();
  System.out.println("****************************************");
  System.out.println("-----------Meeting Table Stock-----------");
  System.out.println("****************************************");
```

1)Continue...



```
for(int i=0;i<Meeting_Tables.length;i++){
    System.out.println();
    System.out.println("-----------Meeting Table "+(i+1)+". Model-----------");
    for(int j=0;j<Meeting_Tables[i].length;j++){
        System.out.println((ctr+1)+". -> Meeting Table "+ (i+1) +".Model "+(j+1)+".Color : "+Meeting_Tables[i][j]);
        ctr++;
    }}
System.out.println();
System.out.println("****************************************");
System.out.println("-----------Bookcase Stock-----------");
System.out.println("****************************************");
System.out.println("

for(int i=0;i<Bookcases.length;i++){
    System.out.println((ctr+1)+". -> Bookcase "+ (i+1) +".Model : "+Bookcases[i]);
    ctr++;
}
System.out.println();
System.out.println("****************************************");
System.out.println("-----------Office Cabinet Stock-----------");
System.out.println("****************************************");
System.out.println("

for(int i=0;i<Office_Cabinets.length;i++){
    System.out.println((ctr+1)+". -> Office Cabinet "+ (i+1) +".Model : "+Office_Cabinets[i]);
    ctr++;
}}
```

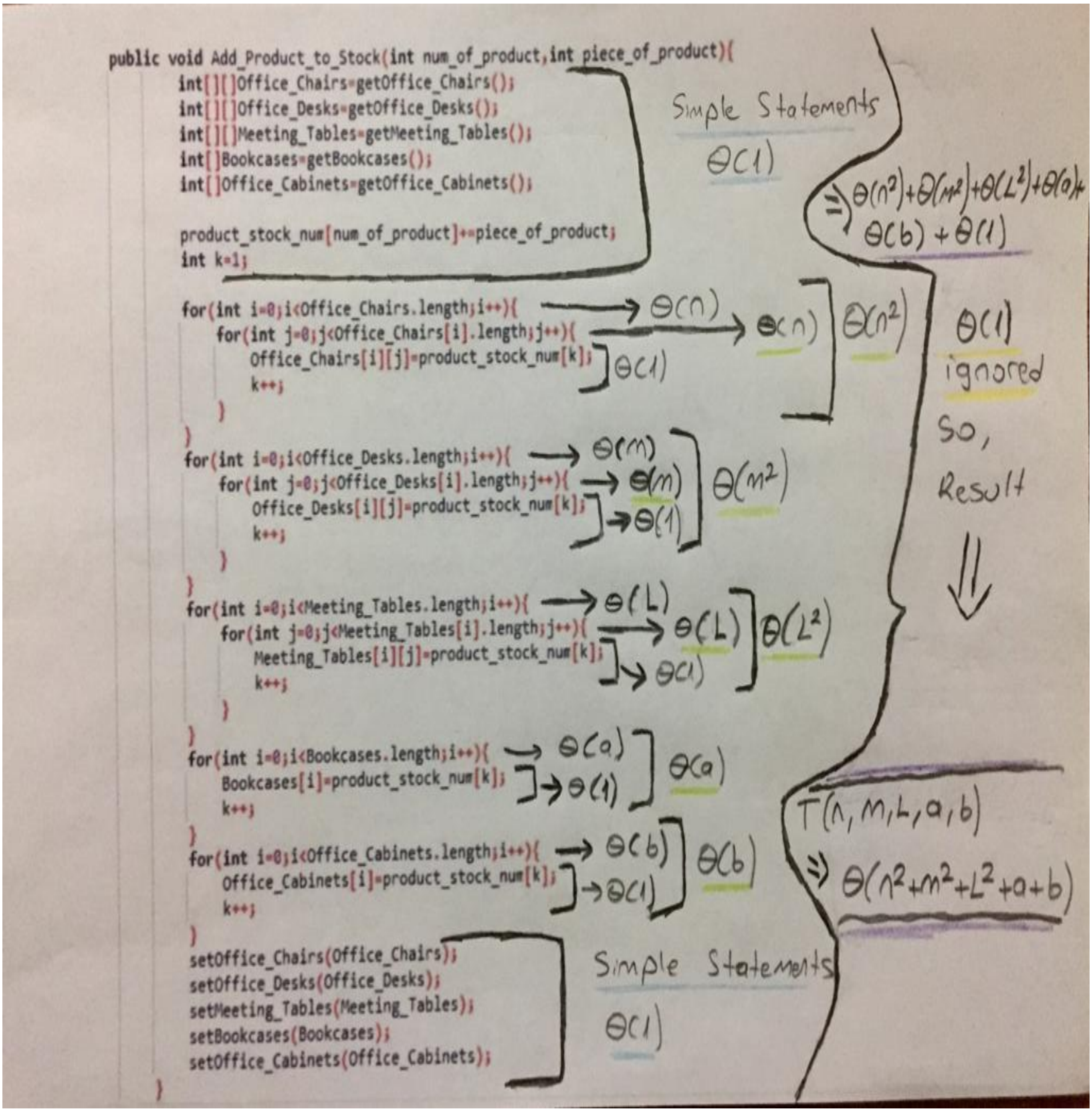
$\Rightarrow \Theta(n^2)+\Theta(m^2)+\Theta(p^2)+\Theta(a)+\Theta(b)+\Theta(1)$

$\Rightarrow \Theta(1)$ ignored

$\Rightarrow T(n,m,p,a,b)=\underline{\Theta(n^2+m^2+p^2+a+b)}$

2)Add/Remove Product

Add Product



```
public void Add_Product_to_Stock(int num_of_product,int piece_of_product){
        int[][]Office_Chairs=getOffice_Chairs();
        int[][]Office_Desks=getOffice_Desks();
        int[][]Meeting_Tables=getMeeting_Tables();
        int[]Bookcases=getBookcases();
        int[]Office_Cabinets=getOffice_Cabinets();

        product_stock_num[num_of_product]+=piece_of_product;
        int k=1;

        for(int i=0;i<Office_Chairs.length;i++){
            for(int j=0;j<Office_Chairs[i].length;j++){
                Office_Chairs[i][j]=product_stock_num[k];
                k++;
            }
        }
        for(int i=0;i<Office_Desks.length;i++){
            for(int j=0;j<Office_Desks[i].length;j++){
                Office_Desks[i][j]=product_stock_num[k];
                k++;
            }
        }
        for(int i=0;i<Meeting_Tables.length;i++){
            for(int j=0;j<Meeting_Tables[i].length;j++){
                Meeting_Tables[i][j]=product_stock_num[k];
                k++;
            }
        }
        for(int i=0;i<Bookcases.length;i++){
            Bookcases[i]=product_stock_num[k];
            k++;
        }
        for(int i=0;i<Office_Cabinets.length;i++){
            Office_Cabinets[i]=product_stock_num[k];
            k++;
        }
        setOffice_Chairs(Office_Chairs);
        setOffice_Desks(Office_Desks);
        setMeeting_Tables(Meeting_Tables);
        setBookcases(Bookcases);
        setOffice_Cabinets(Office_Cabinets);
}
```

Simple Statements $\Theta(1)$

$\rightarrow \Theta(n)$ $\rightarrow \Theta(n)$ $\Theta(1)$ $\Theta(n^2)$

$\rightarrow \Theta(m)$ $\rightarrow \Theta(m)$ $\rightarrow \Theta(1)$ $\Theta(m^2)$

$\rightarrow \Theta(L)$ $\rightarrow \Theta(L)$ $\rightarrow \Theta(1)$ $\Theta(L^2)$

$\rightarrow \Theta(a)$ $\rightarrow \Theta(1)$ $\Theta(a)$

$\rightarrow \Theta(b)$ $\rightarrow \Theta(1)$ $\Theta(b)$

Simple Statements $\Theta(1)$

$\Rightarrow \Theta(n^2)+\Theta(m^2)+\Theta(L^2)+\Theta(a)+\Theta(b)+\Theta(1)$

$\Theta(1)$ ignored

So,

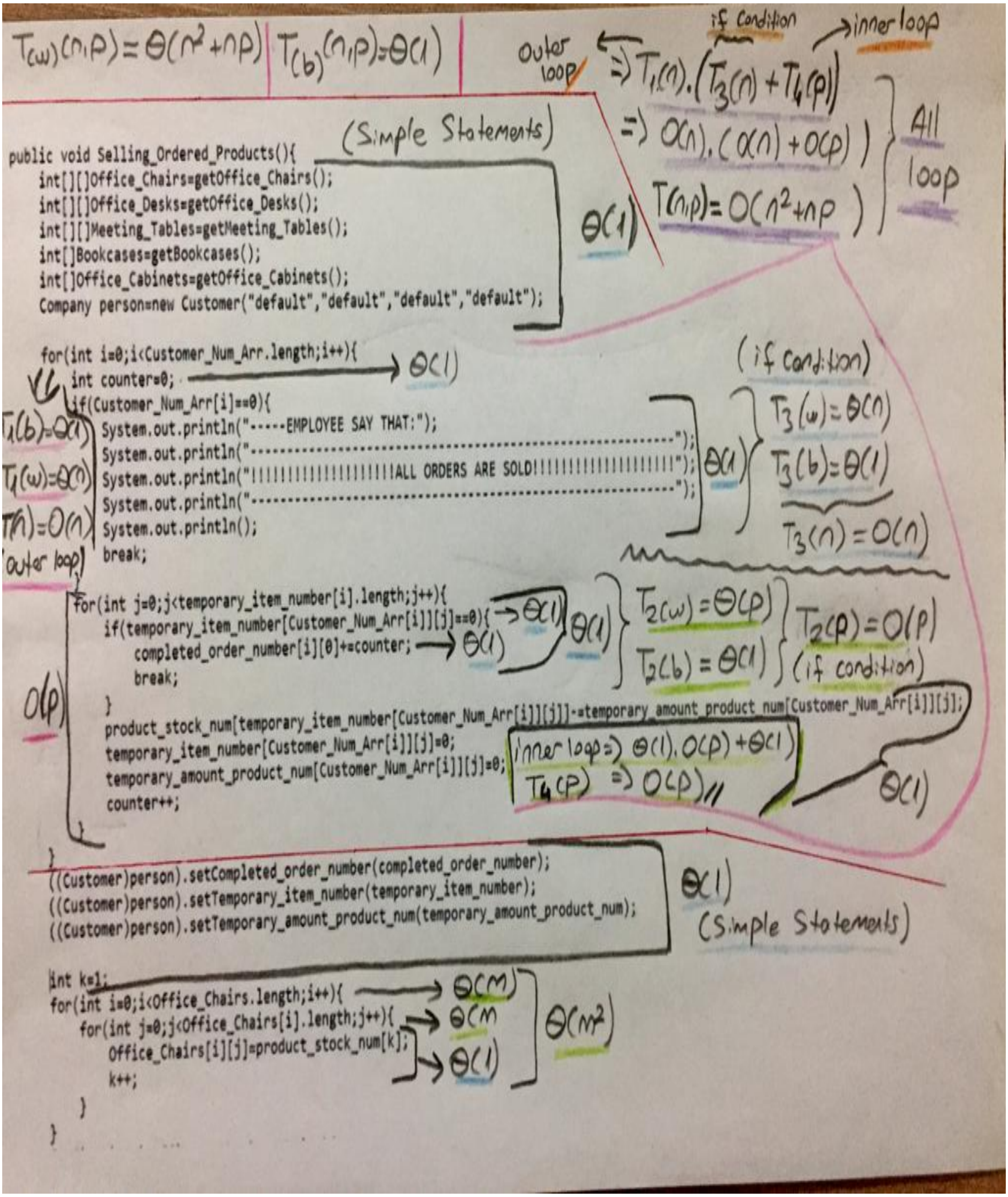
Result

$\Downarrow$

$T(n,m,L,a,b)$

$\Rightarrow \Theta(n^2+m^2+L^2+a+b)$

Remove Product



$T_{(w)}(n,p)=\Theta(n^2+np)$ | $T_{(b)}(n,p)=\Theta(1)$

outer loop

if condition

inner loop

$\Rightarrow T_1(n).(T_3(n)+T_4(p))$

$\Rightarrow O(n).(O(n)+O(p))$

$T(n,p)=O(n^2+np)$

All loop

(Simple Statements)

$\Theta(1)$

```
public void Selling_Ordered_Products(){
    int[][]Office_Chairs=getOffice_Chairs();
    int[][]Office_Desks=getOffice_Desks();
    int[][]Meeting_Tables=getMeeting_Tables();
    int[]Bookcases=getBookcases();
    int[]Office_Cabinets=getOffice_Cabinets();
    Company person=new Customer("default","default","default","default");

    for(int i=0;i<Customer_Num_Arr.length;i++){
        int counter=0;
        if(Customer_Num_Arr[i]==0){
            System.out.println("-----EMPLOYEE SAY THAT:");
            System.out.println("------------------------------------------------------------");
            System.out.println("!!!!!!!!!!!!!!!!!!!!!ALL ORDERS ARE SOLD!!!!!!!!!!!!!!!!!!!!!");
            System.out.println("------------------------------------------------------------");
            System.out.println();
            break;
        }
        for(int j=0;j<temporary_item_number[i].length;j++){
            if(temporary_item_number[Customer_Num_Arr[i]][j]==0){
                completed_order_number[i][0]+=counter;
                break;
            }
            product_stock_num[temporary_item_number[Customer_Num_Arr[i]][j]]-=temporary_amount_product_num[Customer_Num_Arr[i]][j];
            temporary_item_number[Customer_Num_Arr[i]][j]=0;
            temporary_amount_product_num[Customer_Num_Arr[i]][j]=0;
            counter++;
        }
    }
    ((Customer)person).setCompleted_order_number(completed_order_number);
    ((Customer)person).setTemporary_item_number(temporary_item_number);
    ((Customer)person).setTemporary_amount_product_num(temporary_amount_product_num);

    int k=1;
    for(int i=0;i<Office_Chairs.length;i++){
        for(int j=0;j<Office_Chairs[i].length;j++){
            Office_Chairs[i][j]=product_stock_num[k];
            k++;
        }
    }
```

$\Theta(1)$

$T_1(b)=\Theta(1)$

$T_1(w)=\Theta(n)$

$T(n)=O(n)$

(outer loop)

(if condition)

$\Theta(1)$

$T_3(w)=\Theta(n)$

$T_3(b)=\Theta(1)$

$T_3(n)=O(n)$

$\Theta(1)$

$\Theta(1)$

$\Theta(1)$

$T_2(w)=\Theta(p)$

$T_2(b)=\Theta(1)$

$T_2(p)=O(p)$

(if condition)

$O(p)$

inner loop $\Rightarrow \Theta(1).O(p)+\Theta(1)$

$T_4(p) \Rightarrow O(p)$//

$\Theta(1)$

$\Theta(1)$

(Simple Statements)

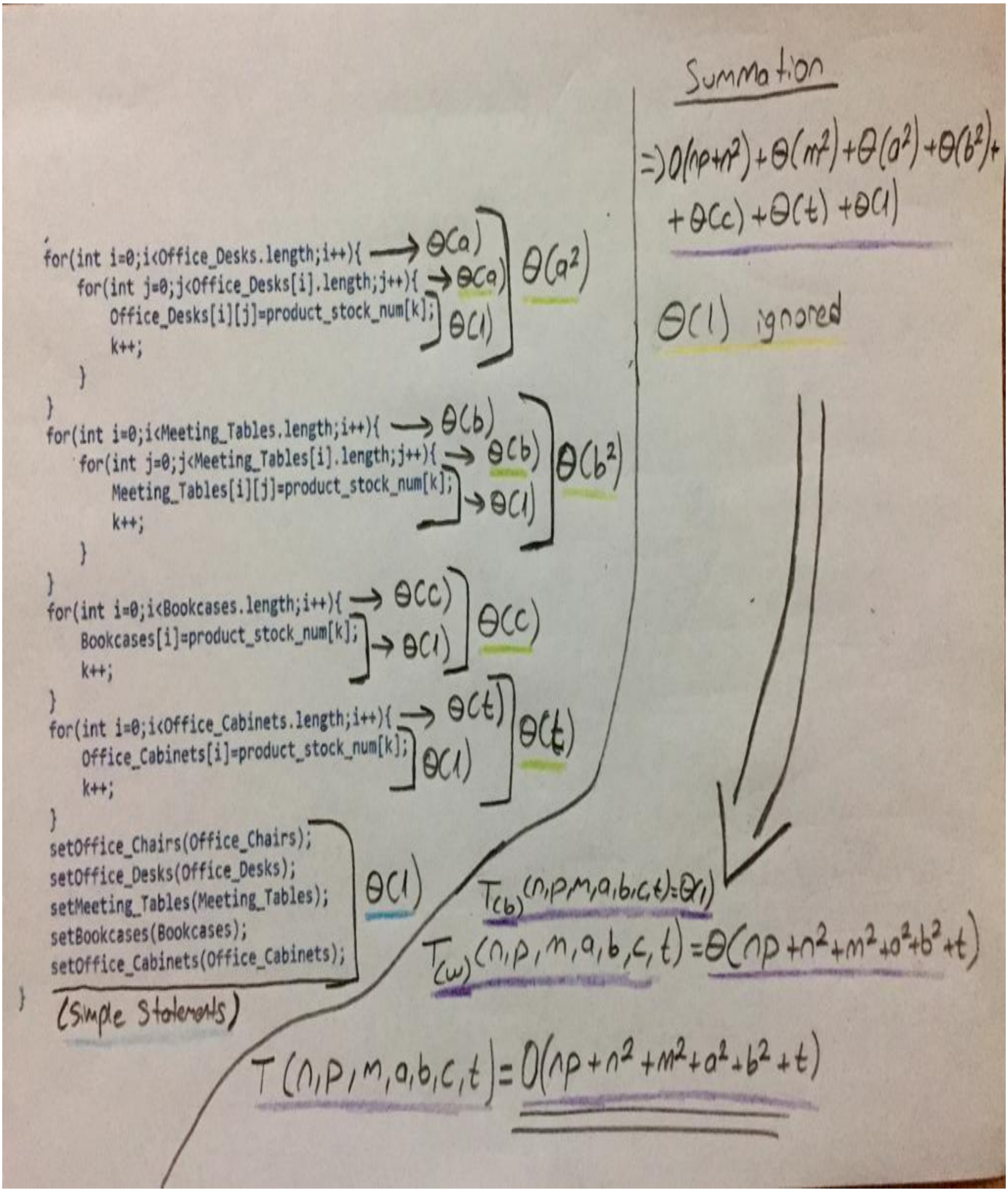
$\Theta(m)$

$\Theta(m$

$\Theta(1)$

$\Theta(m^2)$

Remove Product Continue…



```
for(int i=0;i<Office_Desks.length;i++){
    for(int j=0;j<Office_Desks[i].length;j++){
        Office_Desks[i][j]=product_stock_num[k];
        k++;
    }
}
for(int i=0;i<Meeting_Tables.length;i++){
    for(int j=0;j<Meeting_Tables[i].length;j++){
        Meeting_Tables[i][j]=product_stock_num[k];
        k++;
    }
}
for(int i=0;i<Bookcases.length;i++){
    Bookcases[i]=product_stock_num[k];
    k++;
}
for(int i=0;i<Office_Cabinets.length;i++){
    Office_Cabinets[i]=product_stock_num[k];
    k++;
}
setOffice_Chairs(Office_Chairs);
setOffice_Desks(Office_Desks);
setMeeting_Tables(Meeting_Tables);
setBookcases(Bookcases);
setOffice_Cabinets(Office_Cabinets);
}
```

$\rightarrow \Theta(a)$, $\rightarrow \Theta(a)$, $\Theta(1)$ — $\Theta(a^2)$

$\rightarrow \Theta(b)$, $\rightarrow \Theta(b)$, $\rightarrow \Theta(1)$ — $\Theta(b^2)$

$\rightarrow \Theta(c)$, $\rightarrow \Theta(1)$ — $\Theta(c)$

$\rightarrow \Theta(t)$, $\Theta(1)$ — $\Theta(t)$

$\Theta(1)$

(Simple Statements)

Summation

$$\Rightarrow O(np+n^2)+\Theta(m^2)+\Theta(a^2)+\Theta(b^2)+ +\Theta(c)+\Theta(t)+\Theta(1)$$

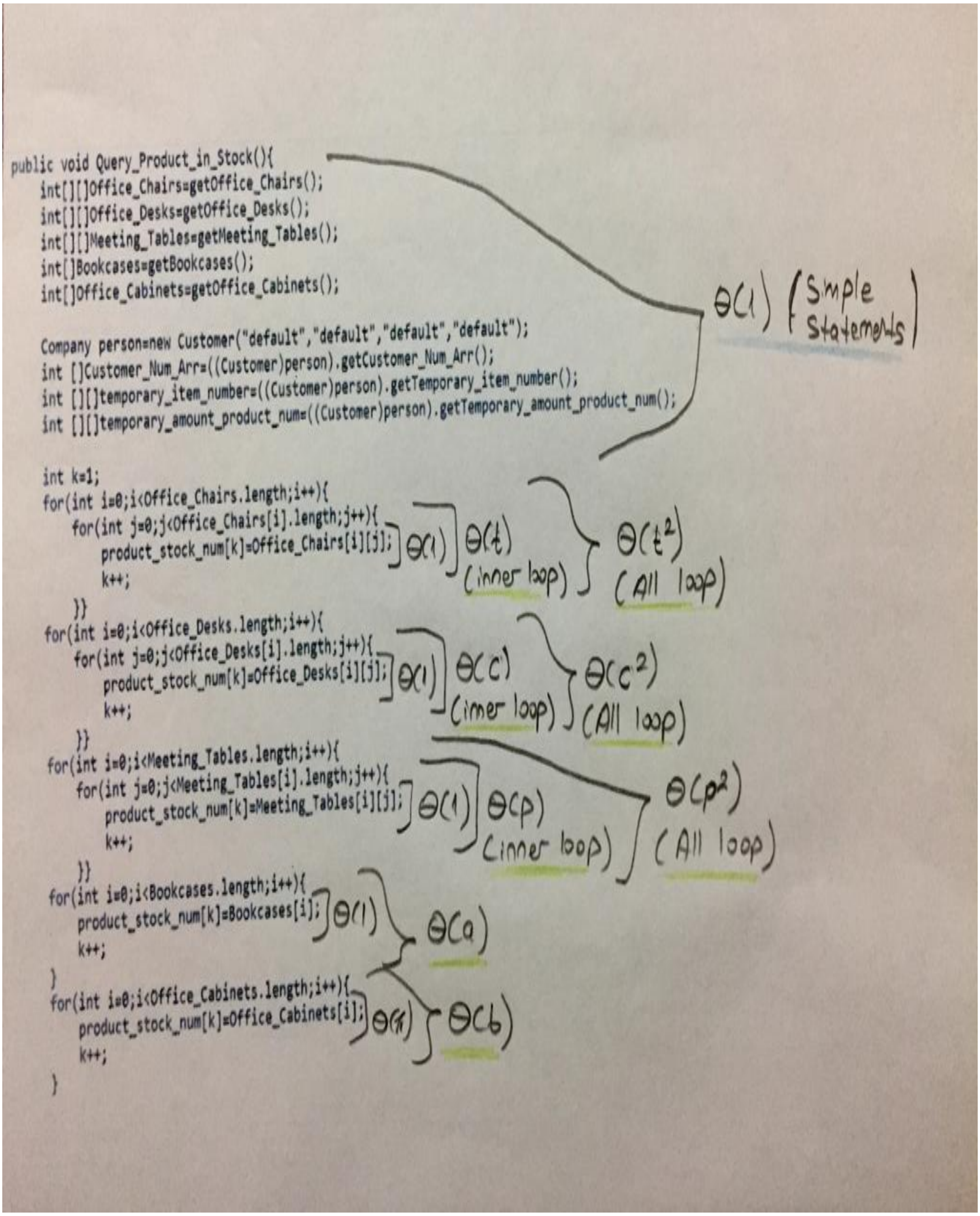
$\Theta(1)$ ignored

$$T_{(b)}(n,p,m,a,b,c,t)=\Theta(1)$$

$$T_{(w)}(n,p,m,a,b,c,t)=\Theta(np+n^2+m^2+a^2+b^2+t)$$

$$T(n,p,m,a,b,c,t)=O(np+n^2+m^2+a^2+b^2+t)$$

3)Querying The Products That Need To Be Supplied

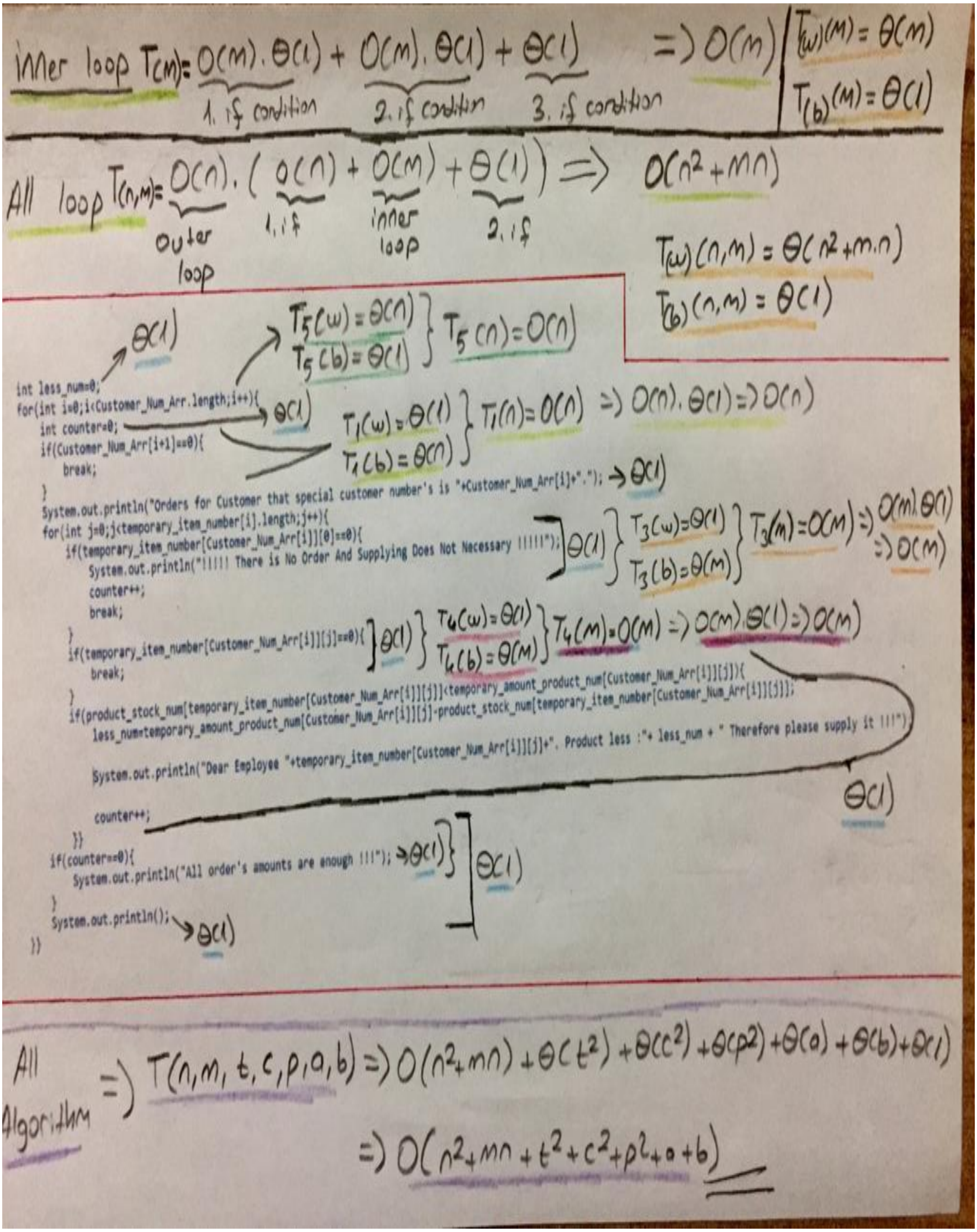


```
public void Query_Product_in_Stock(){
    int[][]Office_Chairs=getOffice_Chairs();
    int[][]Office_Desks=getOffice_Desks();
    int[][]Meeting_Tables=getMeeting_Tables();
    int[]Bookcases=getBookcases();
    int[]Office_Cabinets=getOffice_Cabinets();

    Company person=new Customer("default","default","default","default");
    int []Customer_Num_Arr=((Customer)person).getCustomer_Num_Arr();
    int [][]temporary_item_number=((Customer)person).getTemporary_item_number();
    int [][]temporary_amount_product_num=((Customer)person).getTemporary_amount_product_num();

    int k=1;
    for(int i=0;i<Office_Chairs.length;i++){
        for(int j=0;j<Office_Chairs[i].length;j++){
            product_stock_num[k]=Office_Chairs[i][j];
            k++;
        }}
    for(int i=0;i<Office_Desks.length;i++){
        for(int j=0;j<Office_Desks[i].length;j++){
            product_stock_num[k]=Office_Desks[i][j];
            k++;
        }}
    for(int i=0;i<Meeting_Tables.length;i++){
        for(int j=0;j<Meeting_Tables[i].length;j++){
            product_stock_num[k]=Meeting_Tables[i][j];
            k++;
        }}
    for(int i=0;i<Bookcases.length;i++){
        product_stock_num[k]=Bookcases[i];
        k++;
    }
    for(int i=0;i<Office_Cabinets.length;i++){
        product_stock_num[k]=Office_Cabinets[i];
        k++;
    }
```

3) Continue...



inner loop $T(m) = \underbrace{O(m).\Theta(1)}_{1.\text{ if condition}} + \underbrace{O(m).\Theta(1)}_{2.\text{ if condition}} + \underbrace{\Theta(1)}_{3.\text{ if condition}} \Rightarrow O(m)$

$T_{(w)}(M) = \Theta(m)$

$T_{(b)}(M) = \Theta(1)$

All loop $T(n,m) = \underbrace{O(n)}_{\text{outer loop}} . \left(\underbrace{O(n)}_{1.\text{if}} + \underbrace{O(m)}_{\text{inner loop}} + \underbrace{\Theta(1)}_{2.\text{if}} \right) \Rightarrow O(n^2 + mn)$

$T_{(w)}(n,m) = \Theta(n^2 + m.n)$

$T_{(b)}(n,m) = \Theta(1)$

```
int less_num=0;
for(int i=0;i<Customer_Num_Arr.length;i++){
    int counter=0;
    if(Customer_Num_Arr[i+1]==0){
        break;
    }
    System.out.println("Orders for Customer that special customer number's is "+Customer_Num_Arr[i]+".");
    for(int j=0;j<temporary_item_number[i].length;j++){
        if(temporary_item_number[Customer_Num_Arr[i]][0]==0){
            System.out.println("!!!!! There is No Order And Supplying Does Not Necessary !!!!!");
            counter++;
            break;
        }
        if(temporary_item_number[Customer_Num_Arr[i]][j]==0){
            break;
        }
        if(product_stock_num[temporary_item_number[Customer_Num_Arr[i]][j]]<temporary_amount_product_num[Customer_Num_Arr[i]][j]){
            less_num=temporary_amount_product_num[Customer_Num_Arr[i]][j]-product_stock_num[temporary_item_number[Customer_Num_Arr[i]][j]];
            System.out.println("Dear Employee "+temporary_item_number[Customer_Num_Arr[i]][j]+". Product less :"+ less_num + " Therefore please supply it !!!");
            counter++;
    }}
    if(counter==0){
        System.out.println("All order's amounts are enough !!!");
    }
    System.out.println();
}}
```

$\Theta(1)$

$\left. \begin{array}{l} T_5(w) = \Theta(n) \\ T_5(b) = \Theta(1) \end{array} \right\} T_5(n) = O(n)$

$\Theta(1)$

$\left. \begin{array}{l} T_1(w) = \Theta(1) \\ T_1(b) = \Theta(n) \end{array} \right\} T_1(n) = O(n) \Rightarrow O(n).\Theta(1) \Rightarrow O(n)$

$\rightarrow \Theta(1)$

$\Theta(1) \left\} \begin{array}{l} T_3(w) = \Theta(1) \\ T_3(b) = \Theta(m) \end{array} \right\} T_3(m) = O(M) \Rightarrow O(m).\Theta(1) \Rightarrow O(m)$

$\Theta(1) \left\} \begin{array}{l} T_4(w) = \Theta(1) \\ T_4(b) = \Theta(M) \end{array} \right\} T_4(M) = O(M) \Rightarrow O(m).\Theta(1) \Rightarrow O(m)$

$\Theta(1)$

$\rightarrow \Theta(1)$

$\Theta(1)$

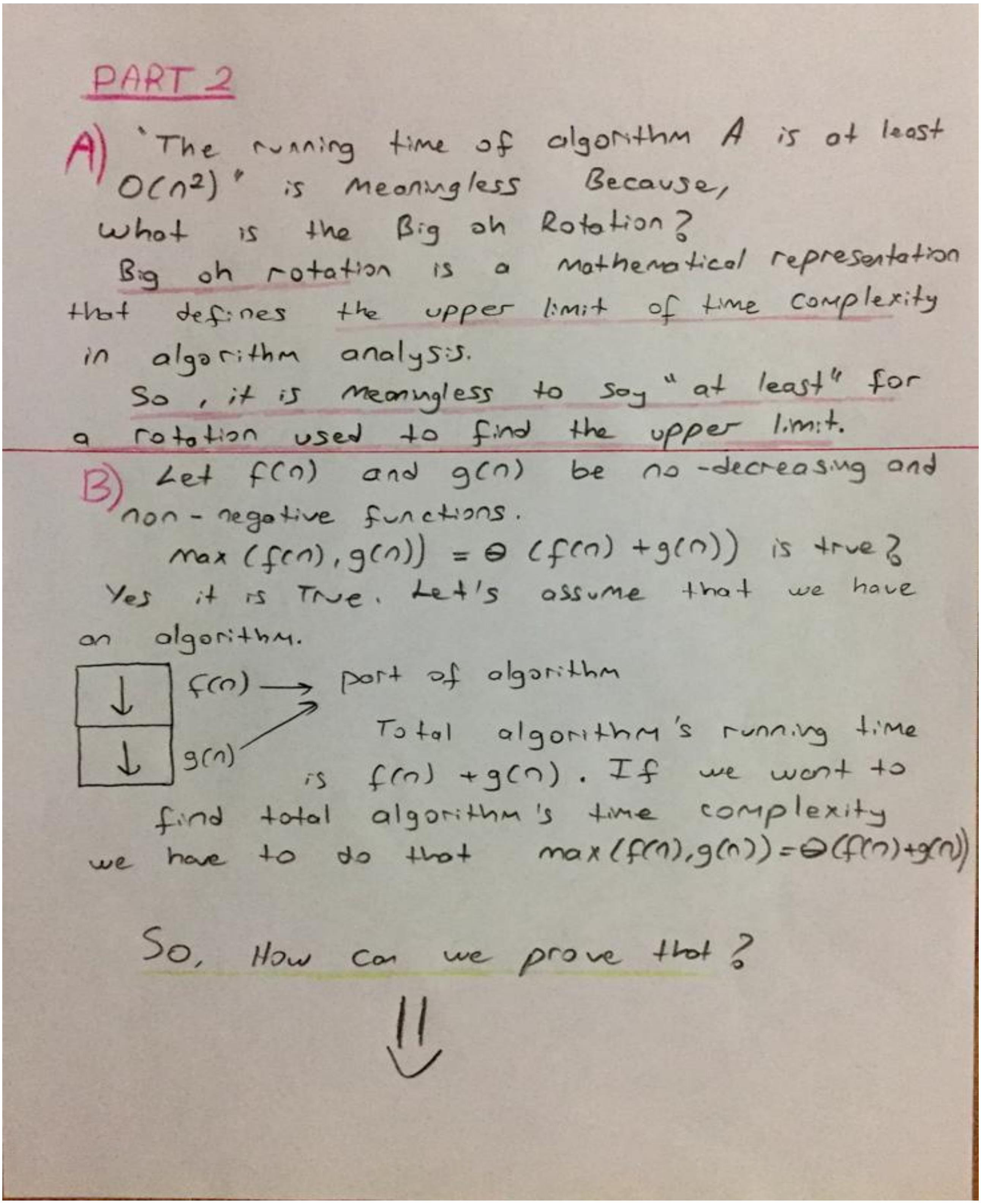
$\rightarrow \Theta(1)$

All Algorithm $\Rightarrow T(n,m,t,c,p,a,b) \Rightarrow O(n^2 + mn) + \Theta(t^2) + \Theta(c^2) + \Theta(p^2) + \Theta(a) + \Theta(b) + \Theta(1)$

$\Rightarrow O(n^2 + mn + t^2 + c^2 + p^2 + a + b)$

PART 2

A) Option Ve B) Option



PART 2

A) 'The running time of algorithm A is at least $O(n^2)$' is meaningless Because, what is the Big oh Rotation?

Big oh rotation is a mathematical representation that defines the upper limit of time complexity in algorithm analysis.

So, it is meaningless to say "at least" for a rotation used to find the upper limit.

B) Let $f(n)$ and $g(n)$ be no-decreasing and non-negative functions.

$\max(f(n), g(n)) = \Theta(f(n) + g(n))$ is true?

Yes it is True. Let's assume that we have an algorithm.

$f(n) \longrightarrow$ part of algorithm

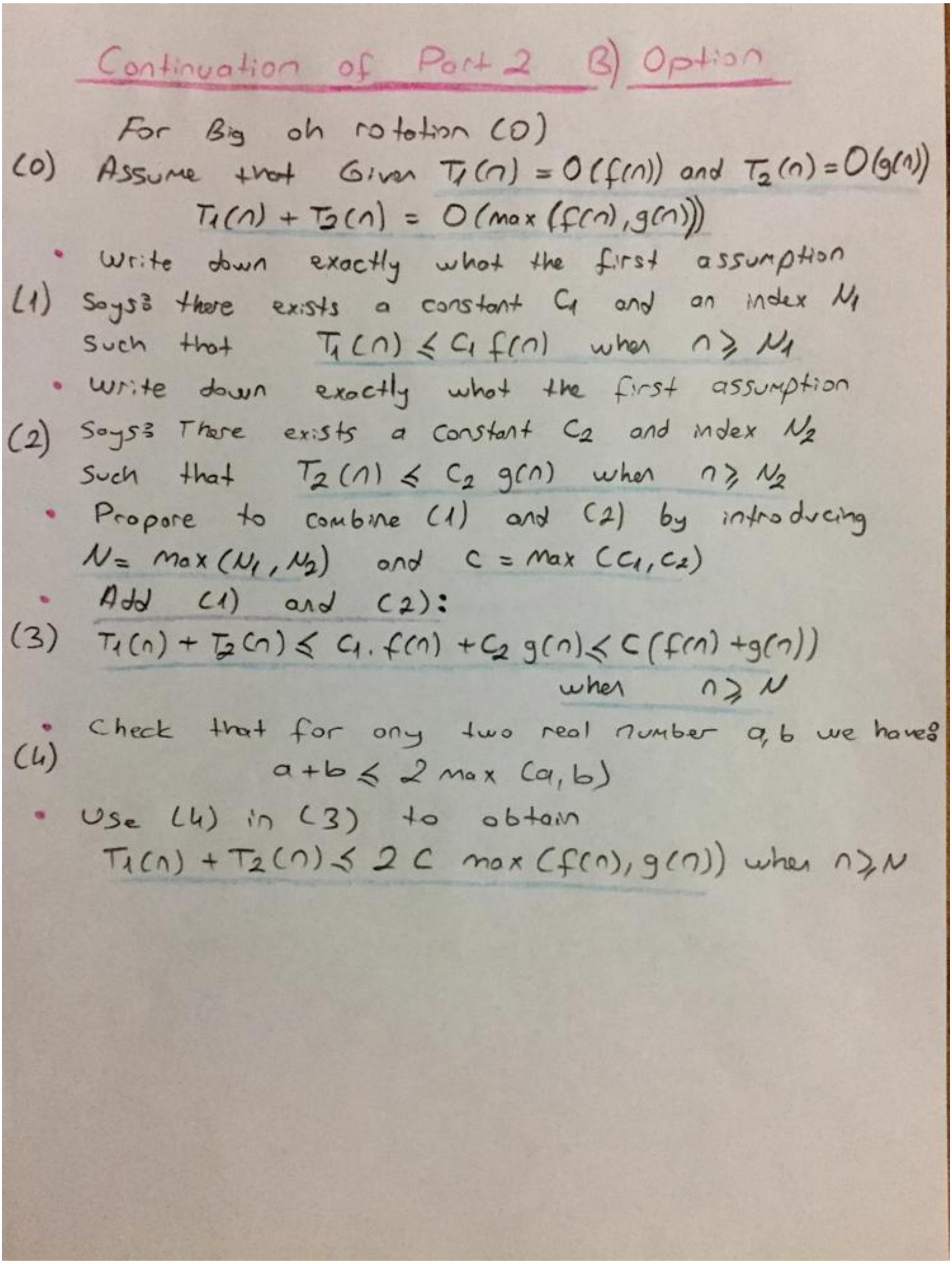
$g(n)$

Total algorithm's running time is $f(n) + g(n)$. If we wont to find total algorithm's time complexity we have to do that $\max(f(n), g(n)) = \Theta(f(n) + g(n))$

So, How can we prove that?

$\Downarrow$

B) Option Continue…



Continuation of Part 2 B) Option

For Big oh notation (O)

(0) Assume that Given $T_1(n) = O(f(n))$ and $T_2(n) = O(g(n))$

$$T_1(n) + T_2(n) = O(\max(f(n), g(n)))$$

- Write down exactly what the first assumption

(1) Says: there exists a constant C_1 and an index N_1 such that $T_1(n) \leq C_1 f(n)$ when $n \geq N_1$

- Write down exactly what the first assumption

(2) Says: There exists a constant C_2 and index N_2 such that $T_2(n) \leq C_2 g(n)$ when $n \geq N_2$

- Propose to combine (1) and (2) by introducing $N = \max(N_1, N_2)$ and $C = \max(C_1, C_2)$
- Add (1) and (2):

(3) $T_1(n) + T_2(n) \leq C_1 . f(n) + C_2 g(n) \leq C(f(n) + g(n))$ when $n \geq N$

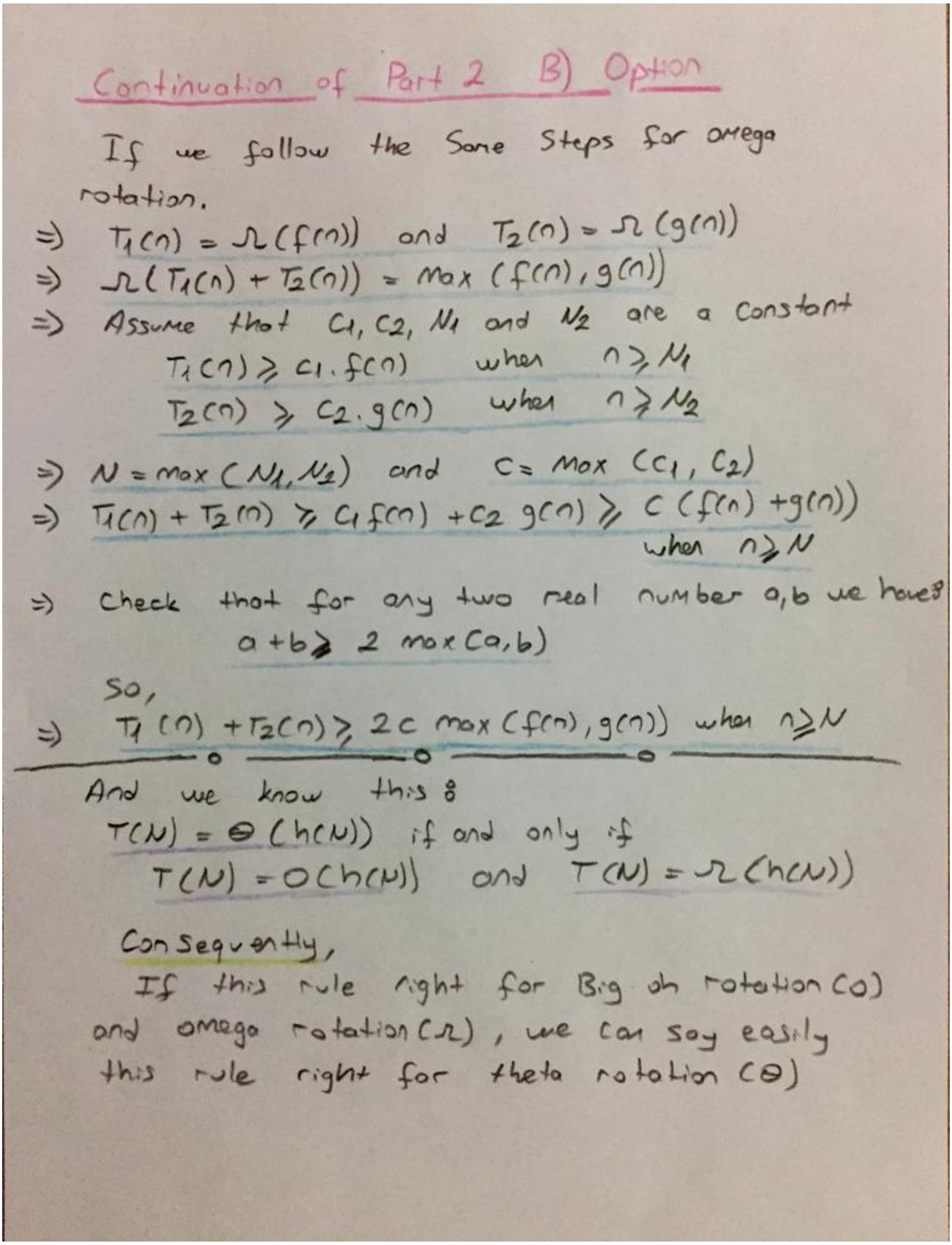
- Check that for any two real number a, b we have:

(4) $$a + b \leq 2 \max(a, b)$$

- Use (4) in (3) to obtain

$$T_1(n) + T_2(n) \leq 2C \max(f(n), g(n)) \text{ when } n \geq N$$

B) Option Continue…



Continuation of Part 2 B) Option

If we follow the Same Steps for omega rotation.

⇒ $T_1(n) = \Omega(f(n))$ and $T_2(n) = \Omega(g(n))$

⇒ $\Omega(T_1(n) + T_2(n)) = max(f(n), g(n))$

⇒ Assume that C_1, C_2, N_1 and N_2 are a constant

$T_1(n) \geq c_1 . f(n)$ when $n \geq N_1$

$T_2(n) \geq c_2 . g(n)$ when $n \geq N_2$

⇒ $N = max(N_1, N_2)$ and $C = Max(C_1, C_2)$

⇒ $T_1(n) + T_2(n) \geq C_1 f(n) + C_2 g(n) \geq C(f(n) + g(n))$ when $n \geq N$

⇒ Check that for any two real number a,b we have:

$a + b \geq 2 max(a,b)$

So,

⇒ $T_1(n) + T_2(n) \geq 2c max(f(n), g(n))$ when $n \geq N$

* * *

And we know this:

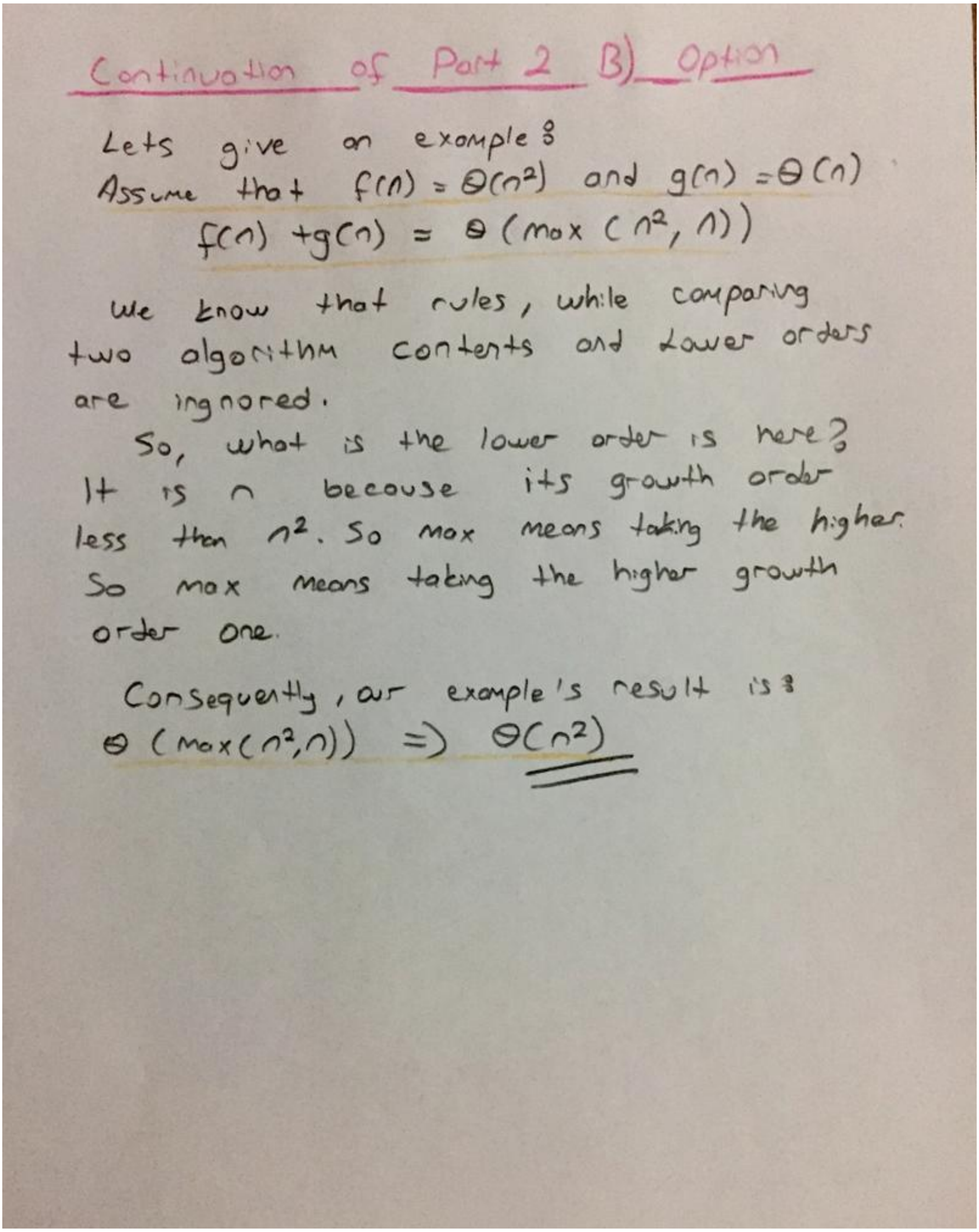
$T(N) = \Theta(h(N))$ if and only if

$T(N) = O(h(N))$ and $T(N) = \Omega(h(N))$

Consequently,

If this rule right for Big oh rotation (O) and omega rotation (Ω), we can say easily this rule right for theta rotation (Θ)

B) Option Continue…



Continuation of Part 2 B) Option

Lets give an example :

Assume that $f(n) = \Theta(n^2)$ and $g(n) = \Theta(n)$

$$f(n) + g(n) = \Theta(\max(n^2, n))$$

We know that rules, while comparing two algorithm contents and Lower orders are ingnored.

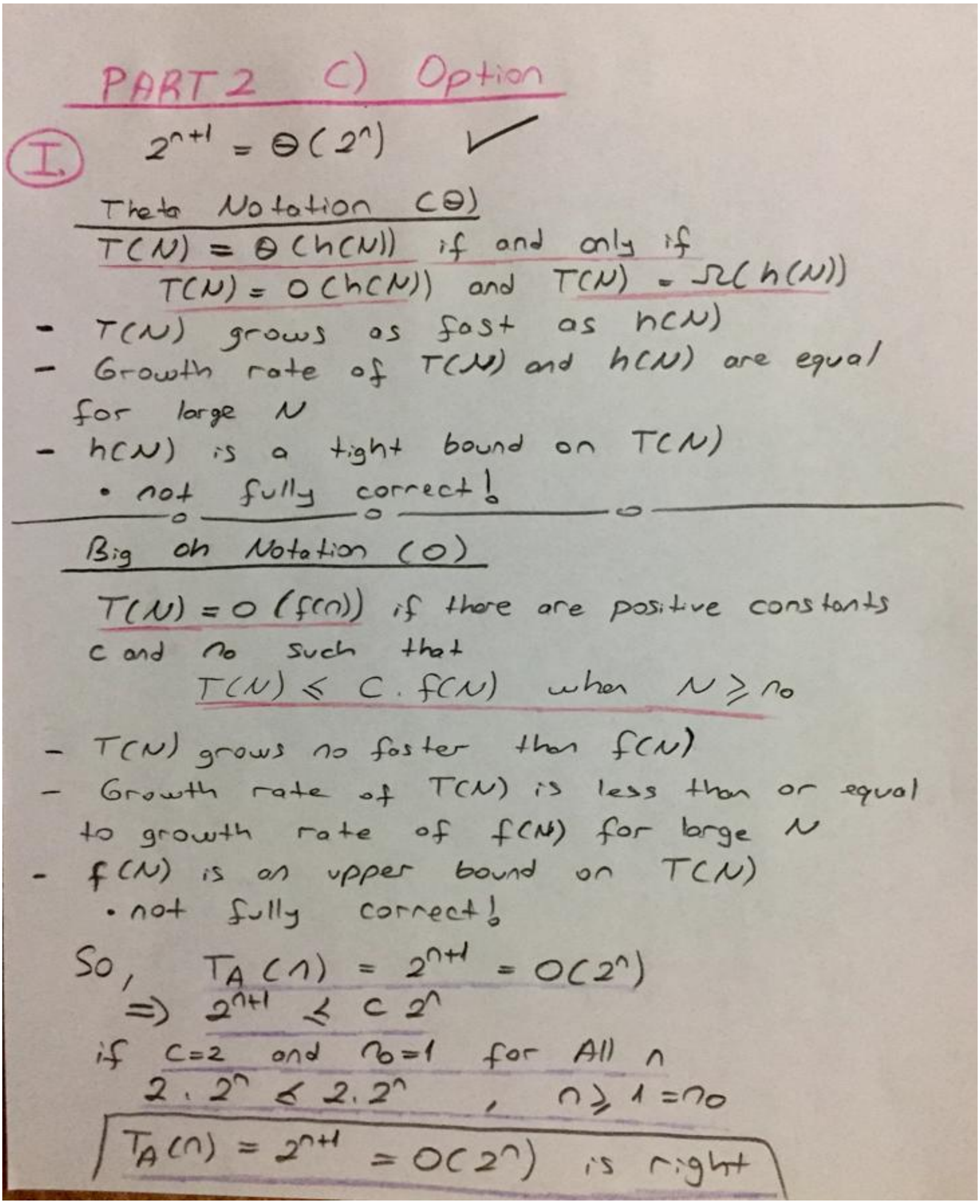
So, what is the lower order is here? It is n becouse its growth order less than n^2. So max means taking the higher. So max means taking the higher growth order one.

Consequently, our example's result is :

$$\Theta(\max(n^2,n)) \Rightarrow \underline{\underline{\Theta(n^2)}}$$

C) Option

1)



PART 2 C) Option

I. $2^{n+1} = \Theta(2^n)$ ✓

Theta Notation (Θ)

$T(N) = \Theta(h(N))$ if and only if

$T(N) = O(h(N))$ and $T(N) = \Omega(h(N))$

- $T(N)$ grows as fast as $h(N)$
- Growth rate of $T(N)$ and $h(N)$ are equal for large N
- $h(N)$ is a tight bound on $T(N)$
  - not fully correct!

Big oh Notation (O)

$T(N) = O(f(n))$ if there are positive constants c and n_0 such that

$T(N) \leq c \cdot f(N)$ when $N \geq n_0$

- $T(N)$ grows no faster than $f(N)$
- Growth rate of $T(N)$ is less than or equal to growth rate of $f(N)$ for large N
- $f(N)$ is an upper bound on $T(N)$
  - not fully correct!

So, $T_A(n) = 2^{n+1} = O(2^n)$

$\Rightarrow 2^{n+1} \leq c\, 2^n$

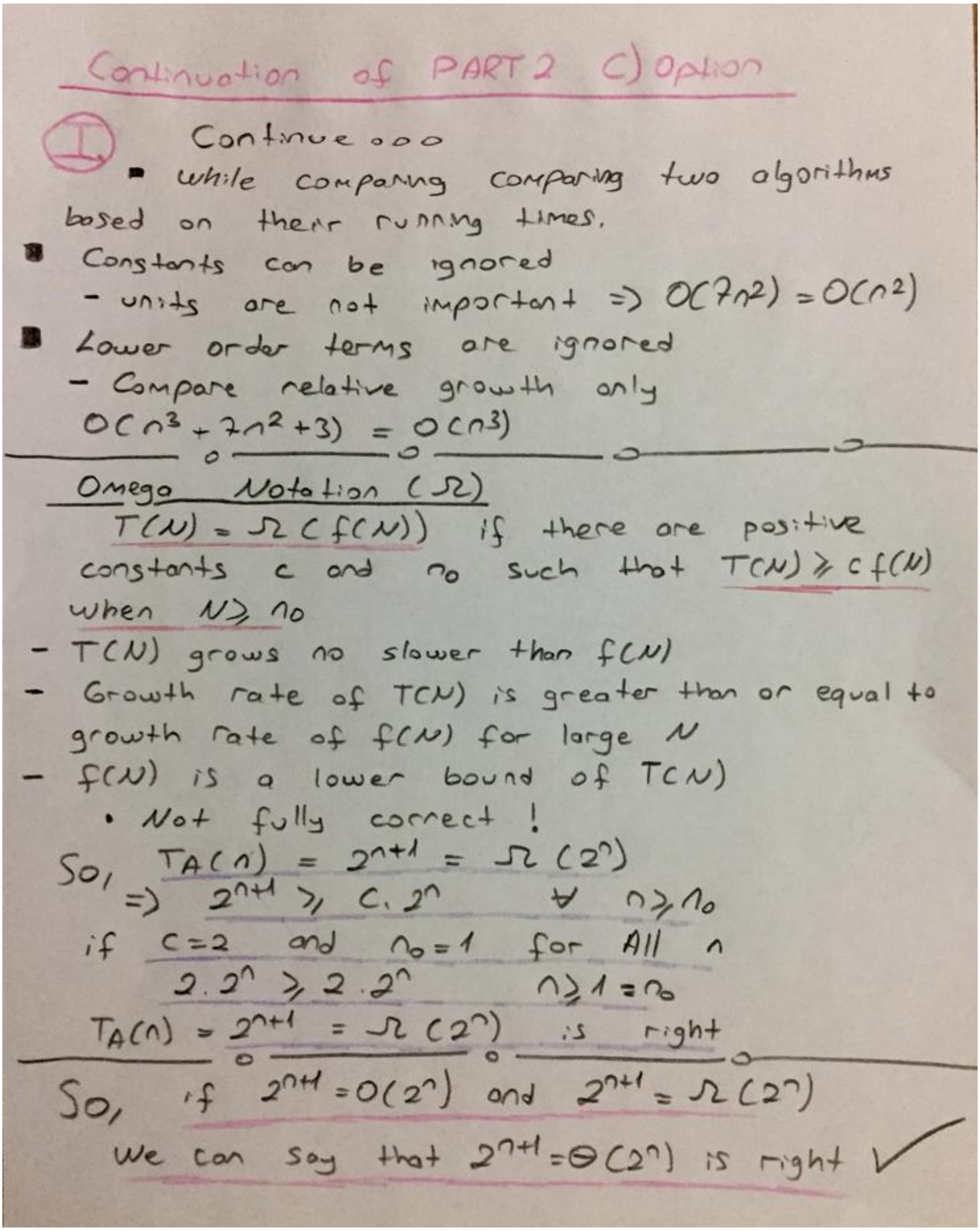
if $c=2$ and $n_0=1$ for All n

$2 \cdot 2^n \leq 2 \cdot 2^n$, $n \geq 1 = n_0$

$\boxed{T_A(n) = 2^{n+1} = O(2^n) \text{ is right}}$

C) Option

1) Continue…



Continuation of PART 2 C) Option

(I) Continue ooo

- while comparing comparing two algorithms based on their running times.
- Constants can be ignored
  - units are not important => $O(7n^2) = O(n^2)$
- Lower order terms are ignored
  - Compare relative growth only

$O(n^3 + 7n^2 + 3) = O(n^3)$

* * *

Omega Notation (Ω)

$T(N) = \Omega(f(N))$ if there are positive constants c and n_0 such that $T(N) \geq c\,f(N)$ when $N \geq n_0$

- $T(N)$ grows no slower than $f(N)$
- Growth rate of $T(N)$ is greater than or equal to growth rate of $f(N)$ for large N
- $f(N)$ is a lower bound of $T(N)$
  - Not fully correct !

So, $T_A(n) = 2^{n+1} = \Omega(2^n)$

$\Rightarrow 2^{n+1} \geq c.2^n \quad \forall \; n \geq n_0$

if $c=2$ and $n_0 = 1$ for All n

$2.2^n \geq 2.2^n \quad n \geq 1 = n_0$

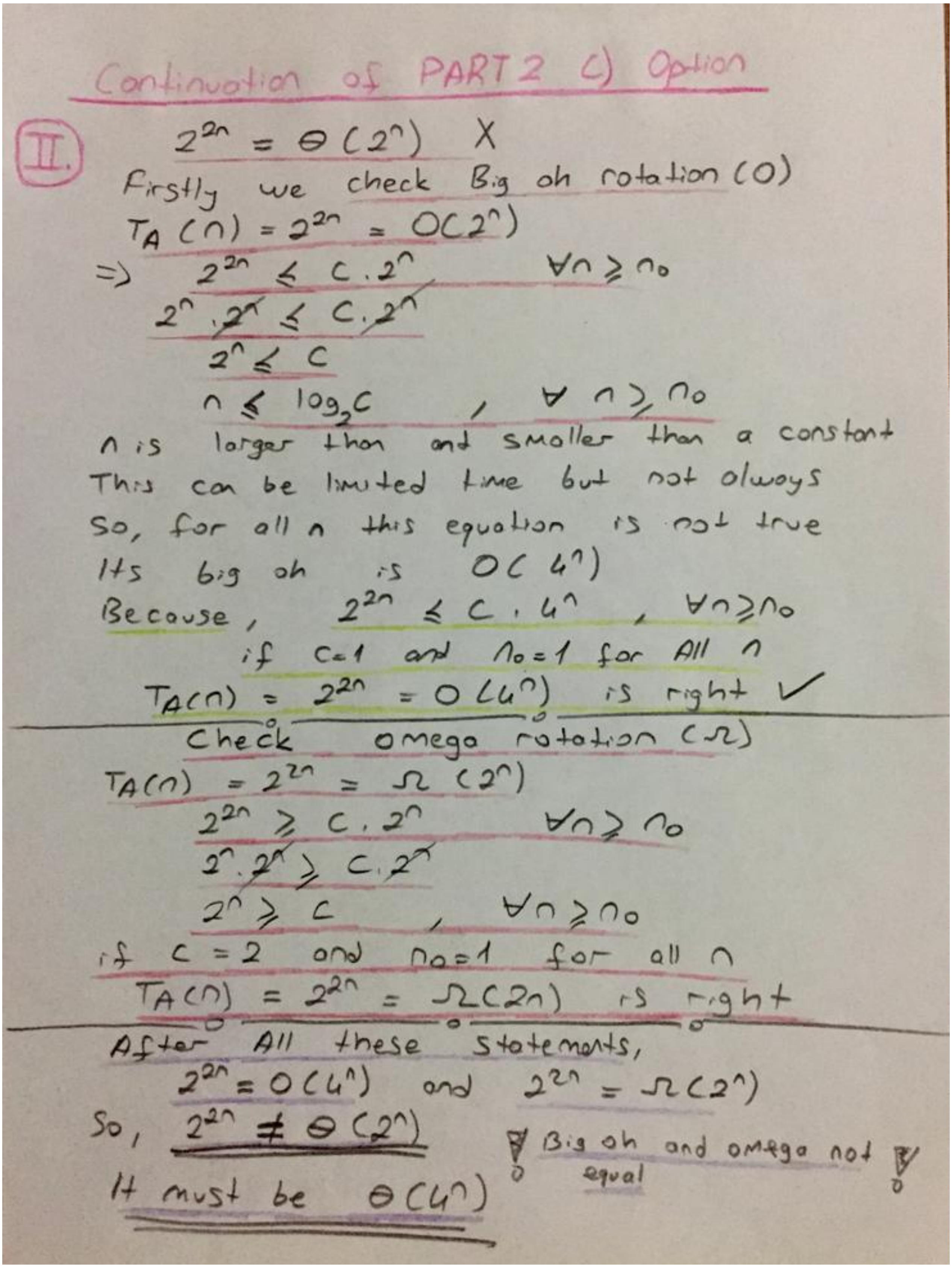
$T_A(n) = 2^{n+1} = \Omega(2^n)$ is right

* * *

So, if $2^{n+1} = O(2^n)$ and $2^{n+1} = \Omega(2^n)$

We can say that $2^{n+1} = \Theta(2^n)$ is right ✓

2)



Continuation of PART 2 c) Option

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

Firstly we check Big oh rotation (O)

$T_A(n) = 2^{2n} = O(2^n)$

$\Rightarrow 2^{2n} \leq c.2^n \qquad \forall n \geq n_0$

$2^n.\cancel{2^n} \leq c.\cancel{2^n}$

$2^n \leq c$

$n \leq \log_2 c \quad , \quad \forall n \geq n_0$

n is larger than and smaller than a constant

This can be limited time but not always

So, for all n this equation is not true

Its big oh is $O(4^n)$

Because, $2^{2n} \leq c.4^n \quad , \quad \forall n \geq n_0$

if $c=1$ and $n_0=1$ for All n

$T_A(n) = 2^{2n} = O(4^n)$ is right ✓

Check omega rotation (Ω)

$T_A(n) = 2^{2n} = \Omega(2^n)$

$2^{2n} \geq c.2^n \qquad \forall n \geq n_0$

$2^n.\cancel{2^n} \geq c.\cancel{2^n}$

$2^n \geq c \quad , \quad \forall n \geq n_0$

if $c = 2$ and $n_0=1$ for all n

$T_A(n) = 2^{2n} = \Omega(2n)$ is right

After All these statements,

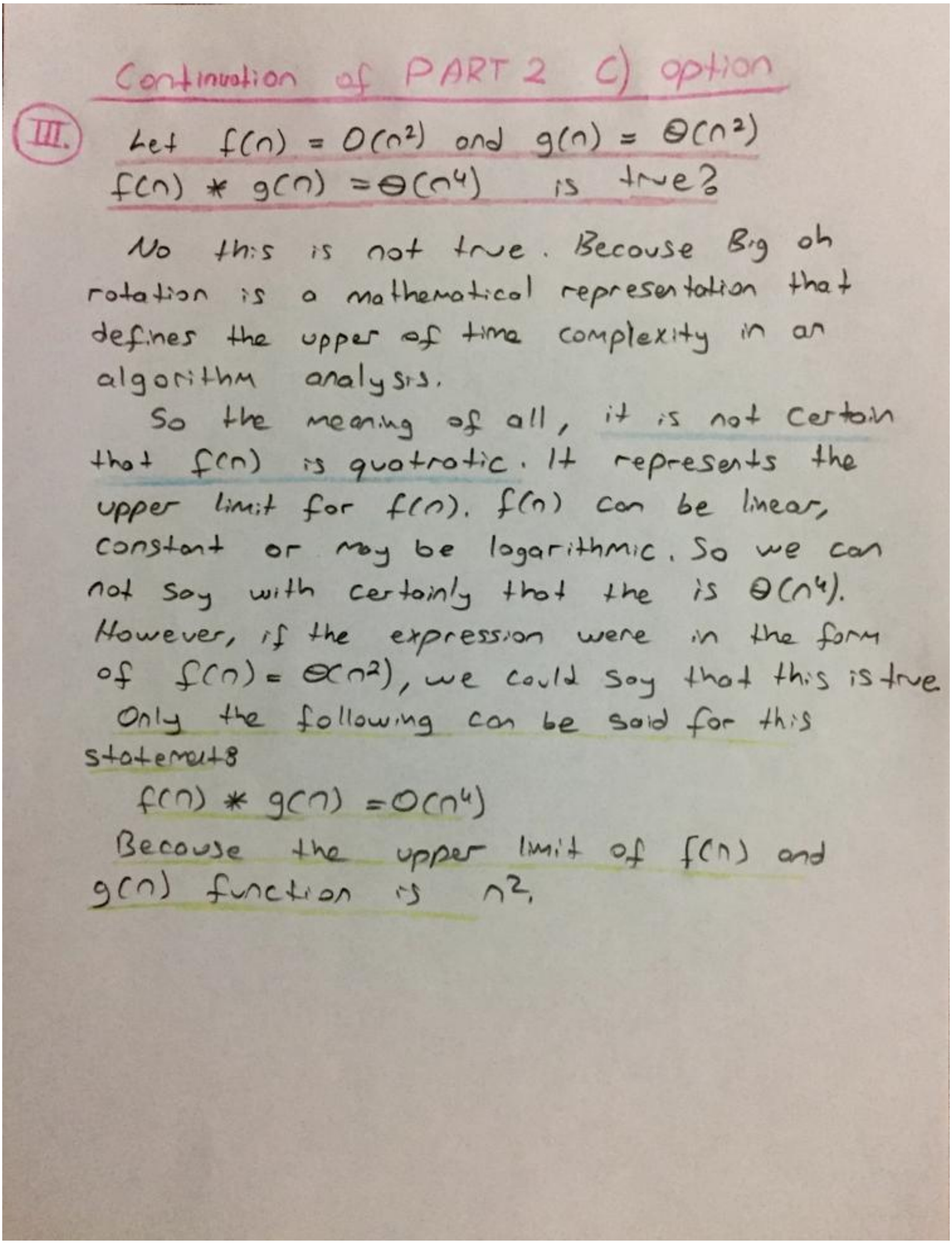
$2^{2n} = O(4^n)$ and $2^{2n} = \Omega(2^n)$

So, $2^{2n} \neq \Theta(2^n)$

It must be $\Theta(4^n)$

! Big oh and omega not equal !

3)



Continuation of PART 2 c) option

III. Let $f(n) = O(n^2)$ and $g(n) = \Theta(n^2)$
$f(n) * g(n) = \Theta(n^4)$ is true?

No this is not true. Becouse Big oh rotation is a mathematical representation that defines the upper of time complexity in an algorithm analysis.

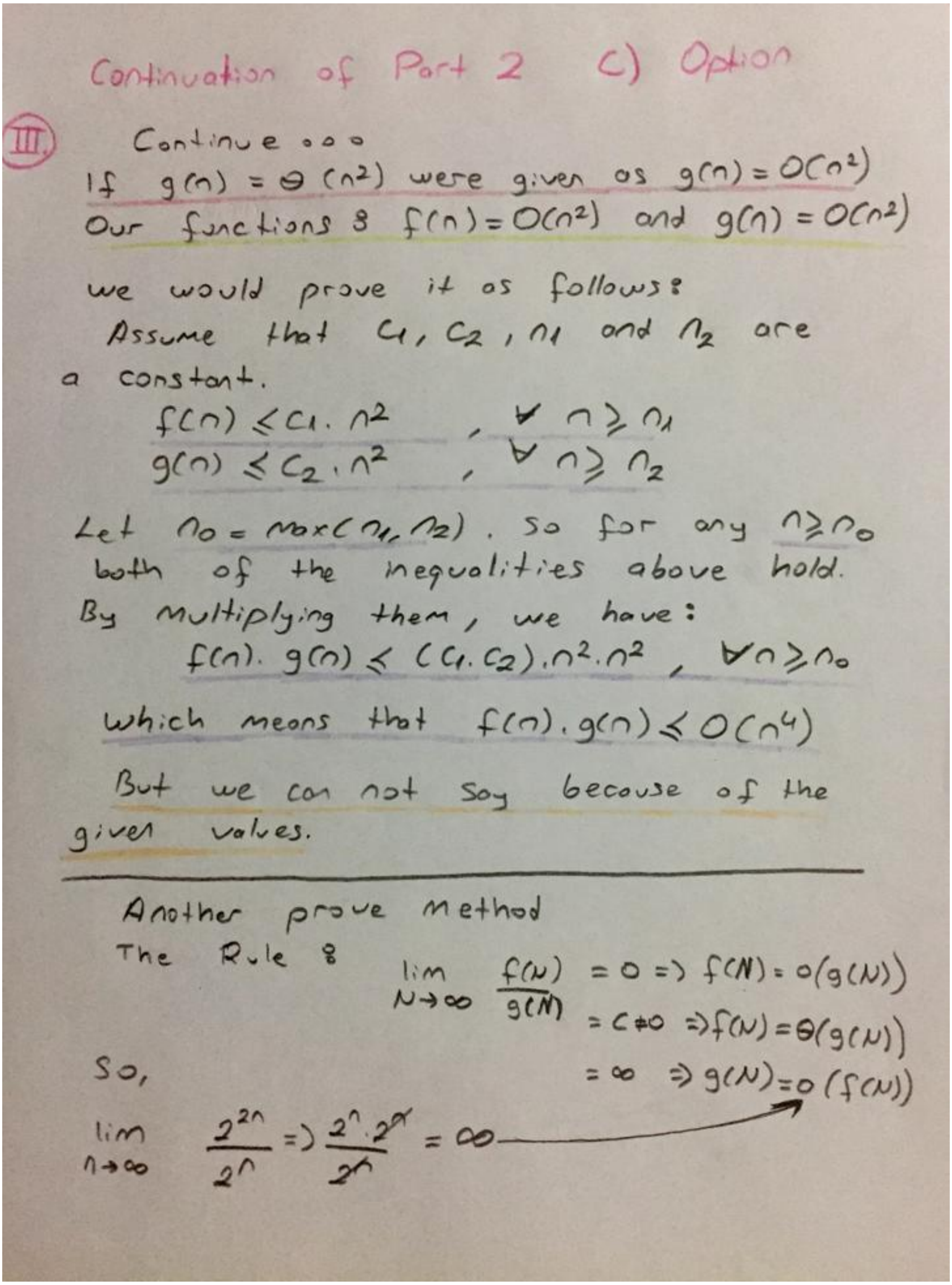
So the meaning of all, it is not certain that $f(n)$ is quotrotic. It represents the upper limit for $f(n)$. $f(n)$ can be linear, constant or may be logarithmic. So we can not say with certainly that the is $\Theta(n^4)$. However, if the expression were in the form of $f(n) = \Theta(n^2)$, we could say that this is true

Only the following can be said for this statement8

$f(n) * g(n) = O(n^4)$

Becouse the upper limit of $f(n)$ and $g(n)$ function is n^2.

3) Continue…



Continuation of Part 2 C) Option

Ⅲ Continue ...

If $g(n) = \Theta(n^2)$ were given as $g(n) = O(n^2)$

Our functions : $f(n) = O(n^2)$ and $g(n) = O(n^2)$

we would prove it as follows:

Assume that c_1, c_2, n_1 and n_2 are a constant.

$$f(n) \leq c_1 \cdot n^2 \quad , \forall n \geq n_1$$
$$g(n) \leq c_2 \cdot n^2 \quad , \forall n \geq n_2$$

Let $n_0 = max(n_1, n_2)$. So for any $n \geq n_0$ both of the inequalities above hold.

By multiplying them, we have:

$$f(n) \cdot g(n) \leq (c_1 \cdot c_2) \cdot n^2 \cdot n^2 \quad , \forall n \geq n_0$$

which means that $f(n) \cdot g(n) \leq O(n^4)$

But we can not say because of the given values.

Another prove method

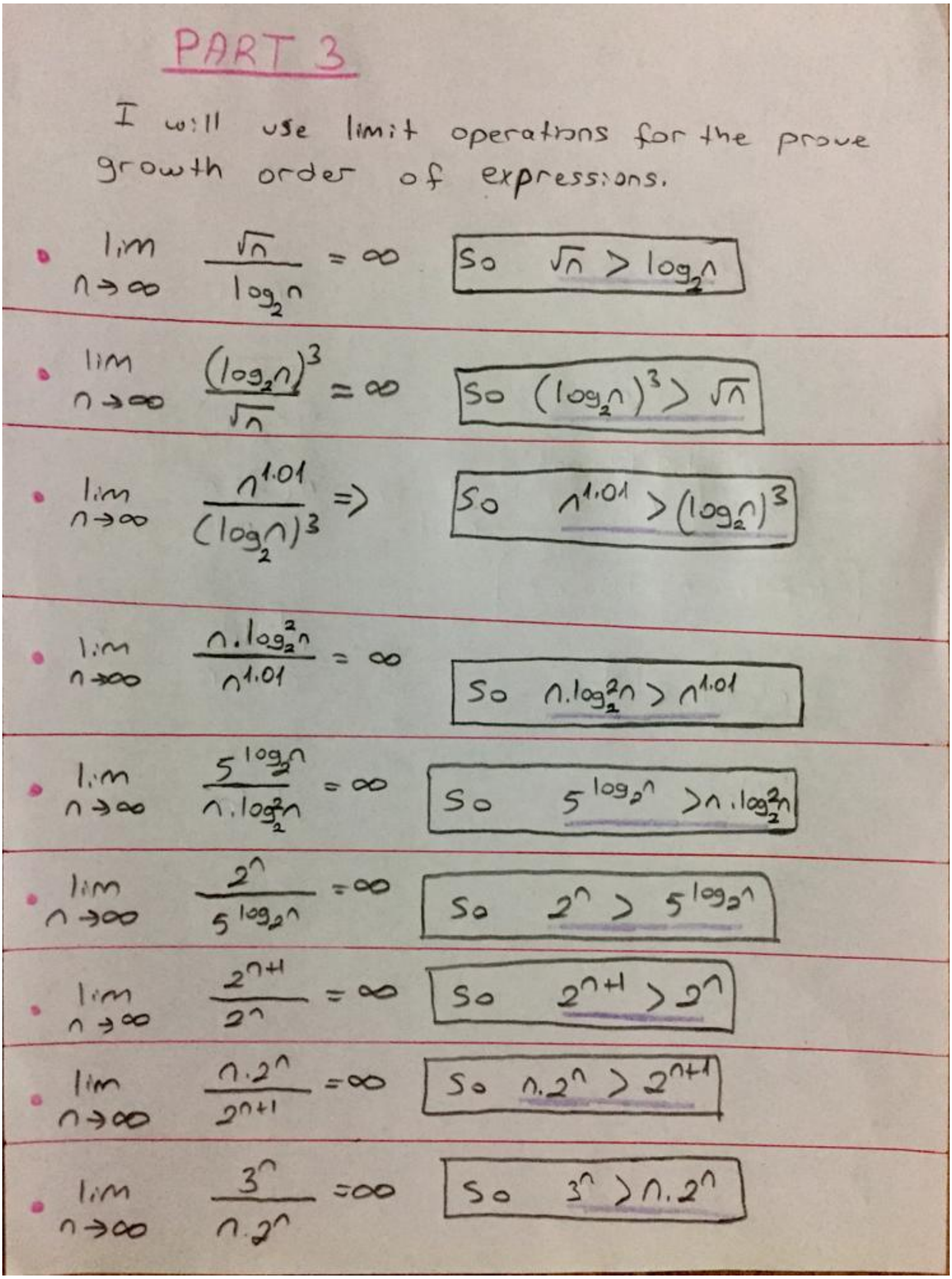
The Rule :

$$\lim_{N \to \infty} \frac{f(N)}{g(N)} \begin{cases} = 0 \Rightarrow f(N) = o(g(N)) \\ = c \neq 0 \Rightarrow f(N) = \Theta(g(N)) \\ = \infty \Rightarrow g(N) = o(f(N)) \end{cases}$$

So,

$$\lim_{n \to \infty} \frac{2^{2n}}{2^n} \Rightarrow \frac{2^n \cdot \cancel{2^n}}{\cancel{2^n}} = \infty$$

PART 3

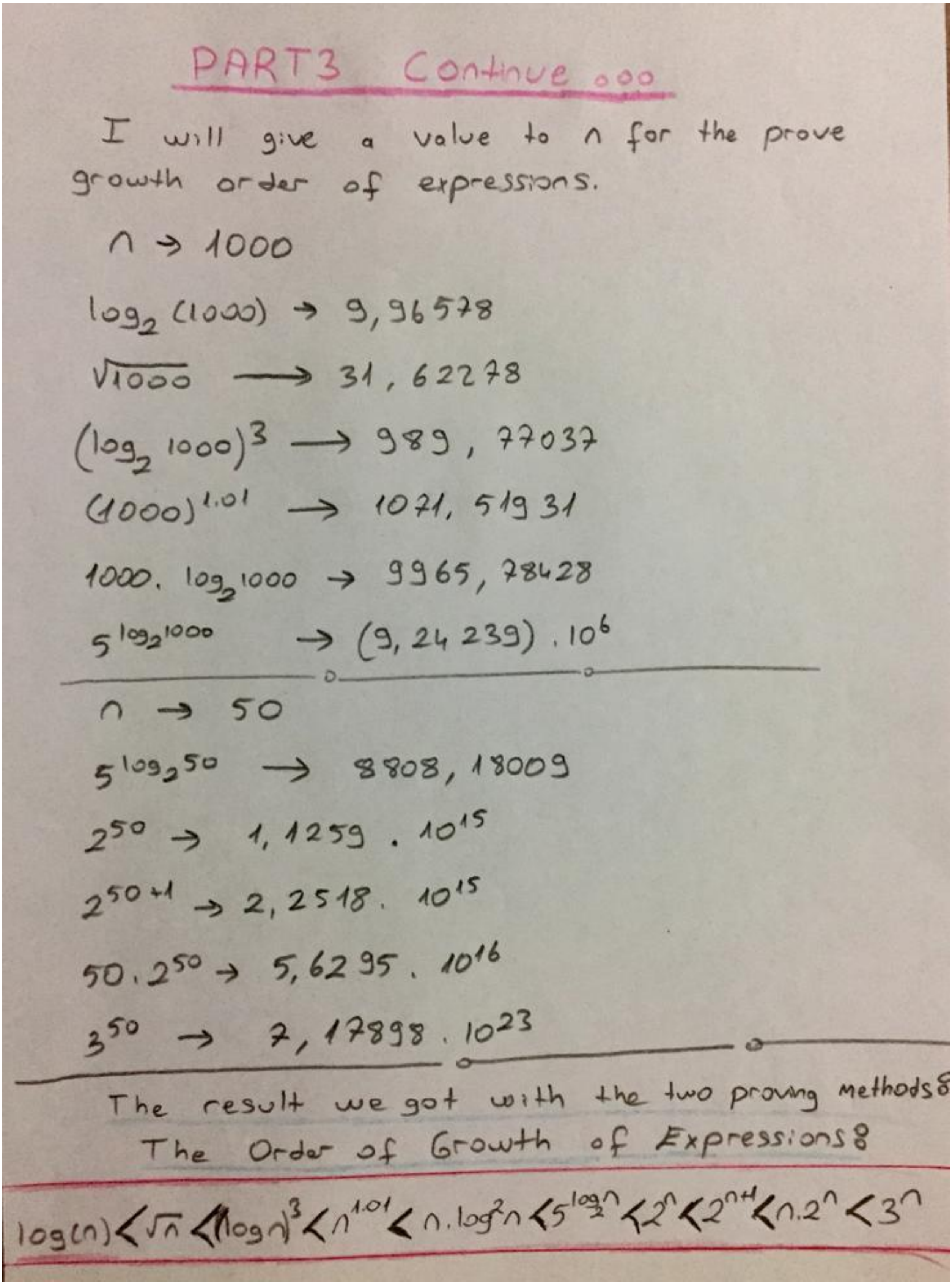


PART 3

I will use limit operations for the prove growth order of expressions.

- $\lim_{n \to \infty} \frac{\sqrt{n}}{\log_2 n} = \infty$ $\boxed{\text{So } \sqrt{n} > \log_2 n}$
- $\lim_{n \to \infty} \frac{(\log_2 n)^3}{\sqrt{n}} = \infty$ $\boxed{\text{So } (\log_2 n)^3 > \sqrt{n}}$
- $\lim_{n \to \infty} \frac{n^{1.01}}{(\log_2 n)^3} \Rightarrow$ $\boxed{\text{So } n^{1.01} > (\log_2 n)^3}$
- $\lim_{n \to \infty} \frac{n.\log_2^2 n}{n^{1.01}} = \infty$ $\boxed{\text{So } n.\log_2^2 n > n^{1.01}}$
- $\lim_{n \to \infty} \frac{5^{\log_2 n}}{n.\log_2^2 n} = \infty$ $\boxed{\text{So } 5^{\log_2 n} > n.\log_2^2 n}$
- $\lim_{n \to \infty} \frac{2^n}{5^{\log_2 n}} = \infty$ $\boxed{\text{So } 2^n > 5^{\log_2 n}}$
- $\lim_{n \to \infty} \frac{2^{n+1}}{2^n} = \infty$ $\boxed{\text{So } 2^{n+1} > 2^n}$
- $\lim_{n \to \infty} \frac{n.2^n}{2^{n+1}} = \infty$ $\boxed{\text{So } n.2^n > 2^{n+1}}$
- $\lim_{n \to \infty} \frac{3^n}{n.2^n} = \infty$ $\boxed{\text{So } 3^n > n.2^n}$

Part 3 Continue...



PART3 Continue ...

I will give a value to n for the prove growth order of expressions.

$n \to 1000$

$\log_2(1000) \to 9,96578$

$\sqrt{1000} \longrightarrow 31,62278$

$(\log_2 1000)^3 \longrightarrow 989,77037$

$(1000)^{1.01} \longrightarrow 1071,51931$

$1000.\log_2 1000 \to 9965,78428$

$5^{\log_2 1000} \to (9,24239).10^6$

$n \to 50$

$5^{\log_2 50} \longrightarrow 8808,18009$

$2^{50} \to 1,1259 . 10^{15}$

$2^{50+1} \to 2,2518 . 10^{15}$

$50.2^{50} \to 5,6295 . 10^{16}$

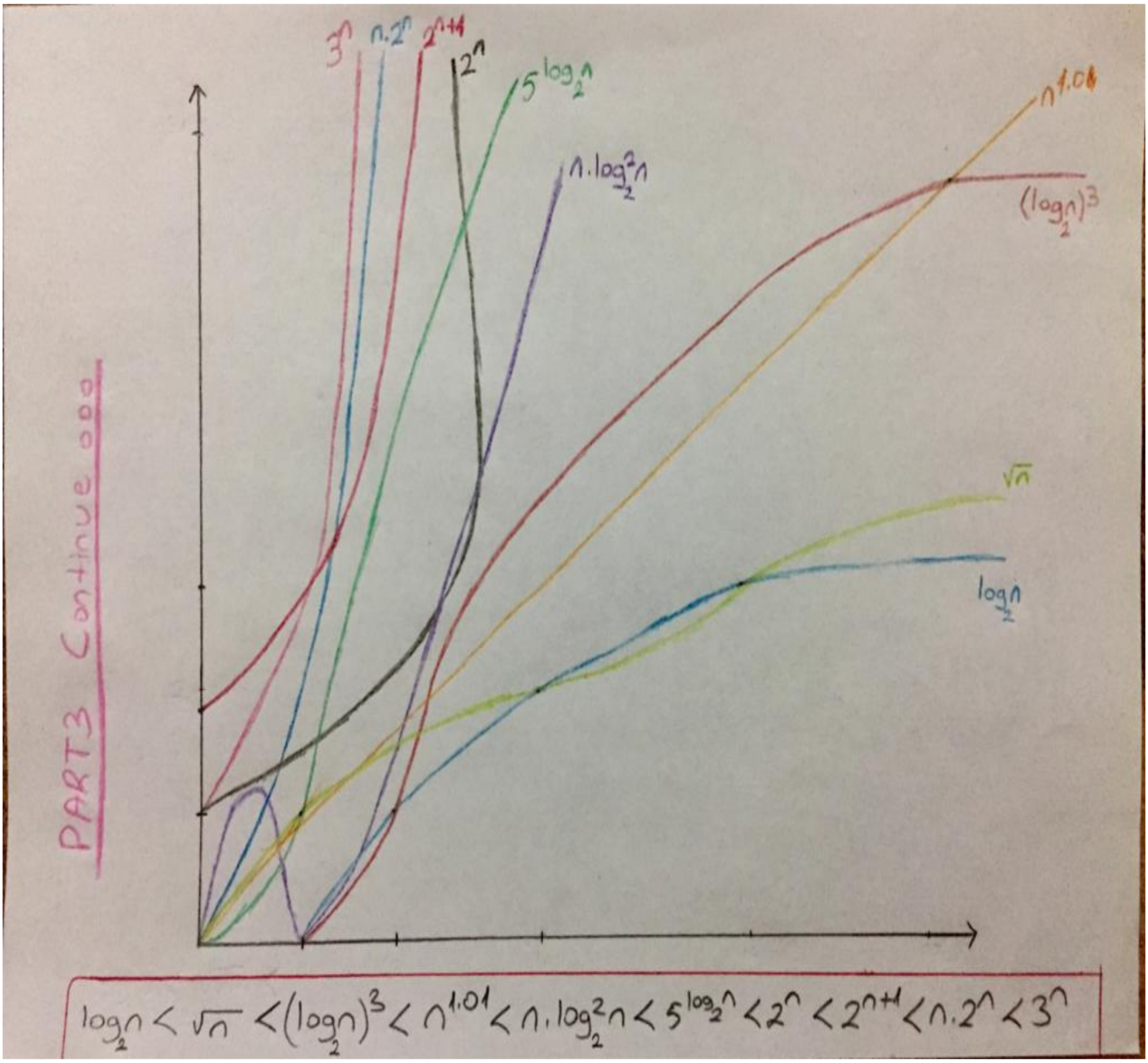
$3^{50} \to 7,17898 . 10^{23}$

The result we got with the two proving methods:

The Order of Growth of Expressions:

$\log(n) < \sqrt{n} < (\log n)^3 < n^{1.01} < n.\log^2 n < 5^{\log_2 n} < 2^n < 2^{n+1} < n.2^n < 3^n$

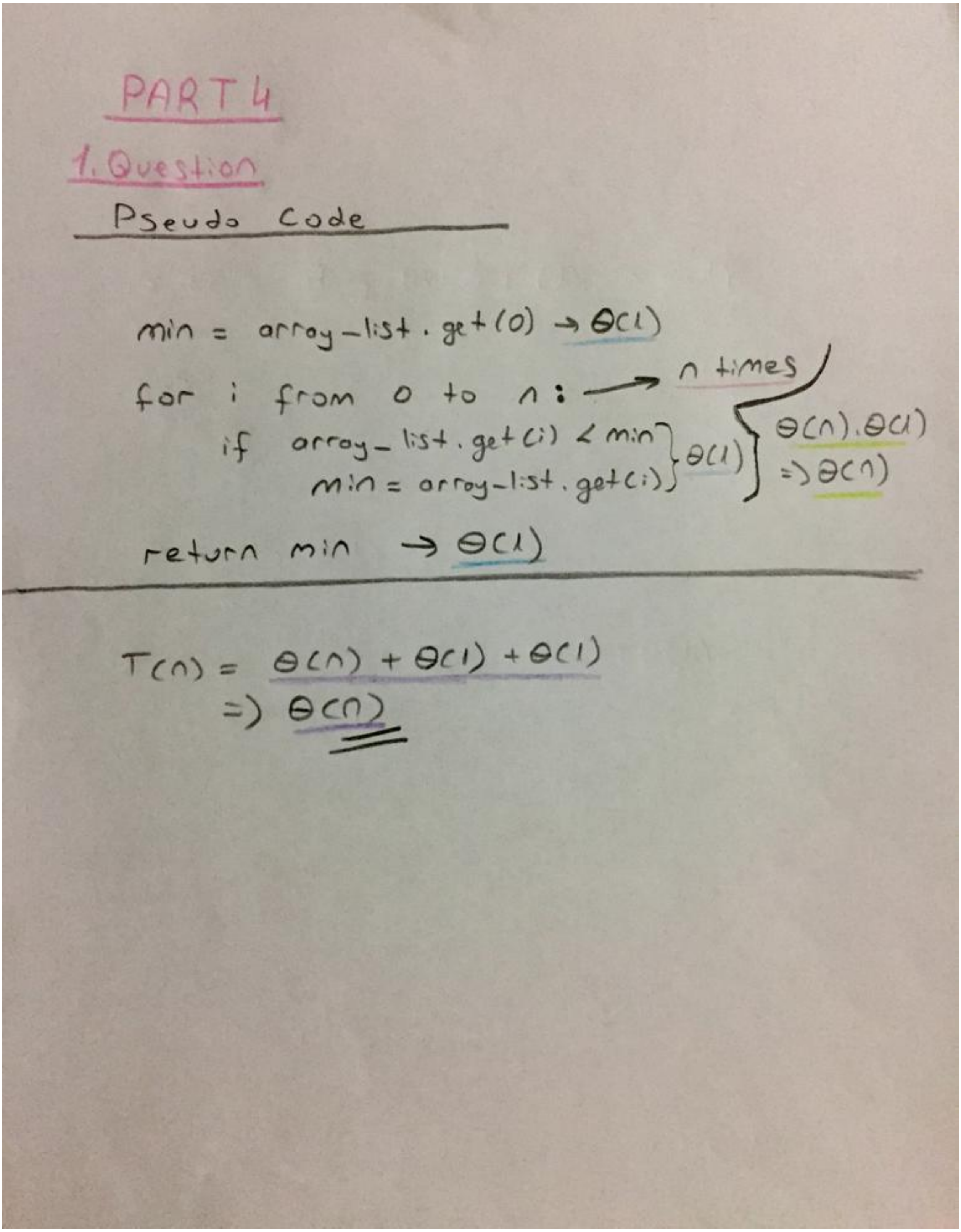
Part 3 Continue...



$$\log_2 n < \sqrt{n} < (\log_2 n)^3 < n^{1.01} < n.\log_2^2 n < 5^{\log_2 n} < 2^n < 2^{n+1} < n.2^n < 3^n$$

PART 4

1. Question



PART 4

1. Question

Pseudo code

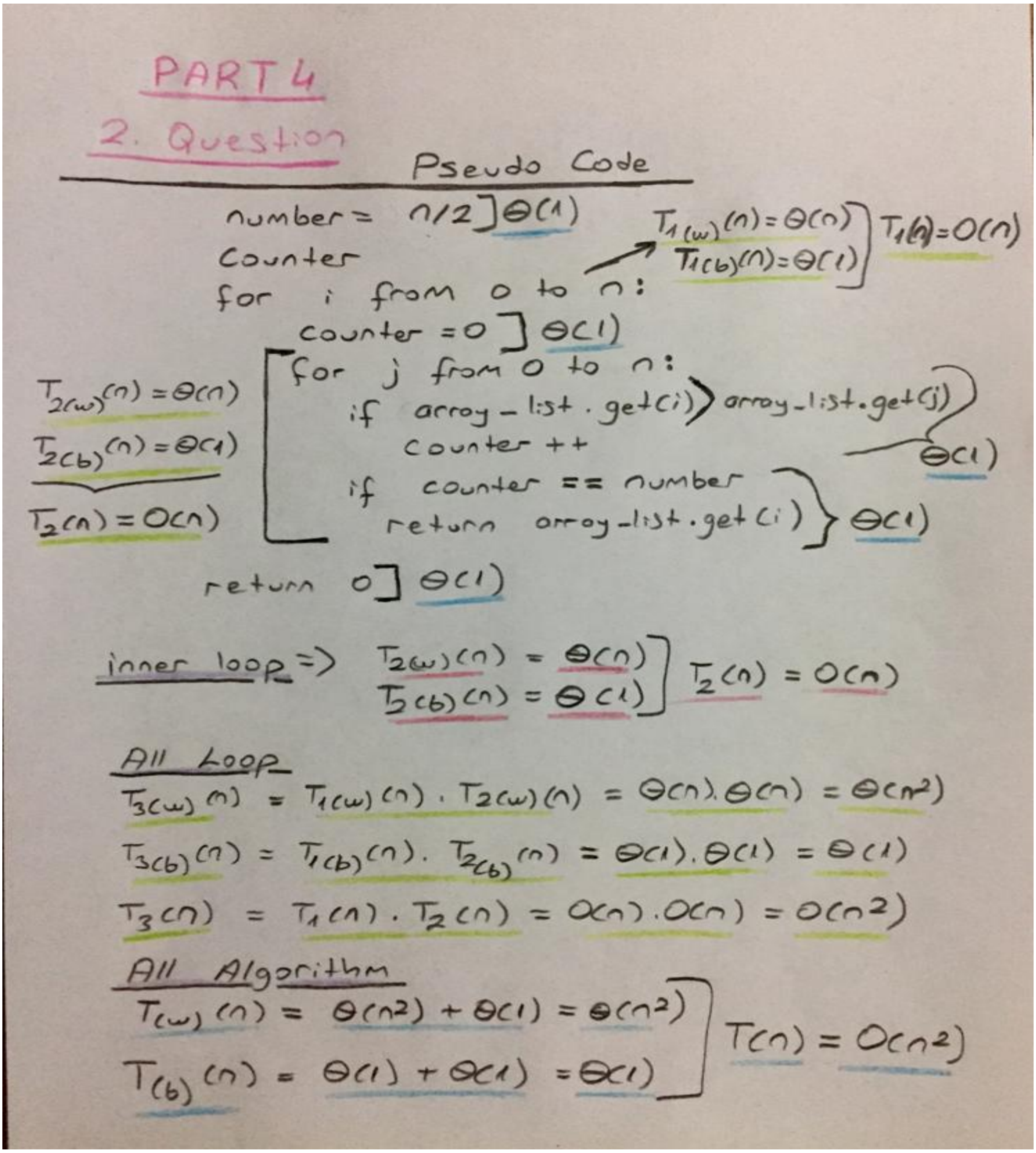
```
min = array_list.get(0) → Θ(1)
for i from 0 to n: → n times
    if array_list.get(i) < min     } Θ(1) } Θ(n).Θ(1)
        min = array_list.get(i)           => Θ(n)
return min → Θ(1)
```

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

$\Rightarrow \Theta(n)$

2. Question

If the given array's index are odd numbers, then the median remains to the right of the middle.The median is greater than half of the elements in the given array.



PART 4

2. Question

Pseudo Code

```
number = n/2 ] $\Theta(1)$
Counter
for i from 0 to n:
    counter = 0 ] $\Theta(1)$
    for j from 0 to n:
        if array-list.get(i) > array-list.get(j)
            counter ++                              $\Theta(1)$
        if counter == number
            return array-list.get(i)                $\Theta(1)$
return 0 ] $\Theta(1)$
```

$T_{1(w)}(n) = \Theta(n)$

$T_{1(b)}(n) = \Theta(1)$

$T_1(n) = O(n)$

$T_{2(w)}(n) = \Theta(n)$

$T_{2(b)}(n) = \Theta(1)$

$T_2(n) = O(n)$

inner loop => $T_{2(w)}(n) = \Theta(n)$, $T_{2(b)}(n) = \Theta(1)$] $T_2(n) = O(n)$

All Loop

$T_{3(w)}(n) = T_{1(w)}(n) . T_{2(w)}(n) = \Theta(n).\Theta(n) = \Theta(n^2)$

$T_{3(b)}(n) = T_{1(b)}(n) . T_{2(b)}(n) = \Theta(1).\Theta(1) = \Theta(1)$

$T_3(n) = T_1(n) . T_2(n) = O(n).O(n) = O(n^2)$

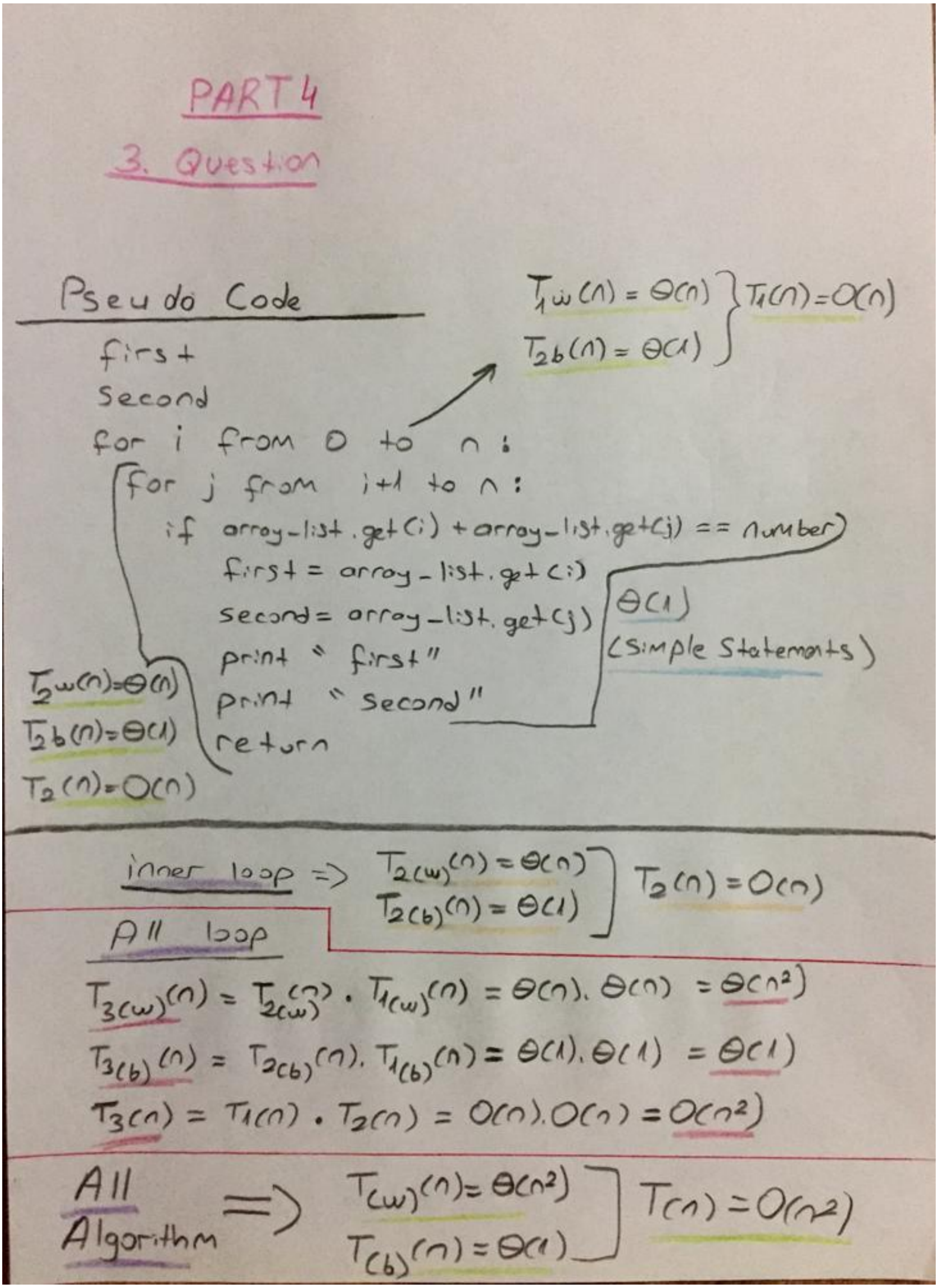
All Algorithm

$T_{(w)}(n) = \Theta(n^2) + \Theta(1) = \Theta(n^2)$

$T_{(b)}(n) = \Theta(1) + \Theta(1) = \Theta(1)$

$T(n) = O(n^2)$

3. Question



PART 4

3. Question

Pseudo Code

```
first
second
for i from 0 to n:
    for j from i+1 to n:
        if array-list.get(i) + array-list.get(j) == number)
            first = array-list.get(i)
            second = array-list.get(j)
            print " first"
            print " second"
            return
```

$T_{1w}(n) = \Theta(n)$

$T_{2b}(n) = \Theta(1)$

$T_1(n) = O(n)$

$\Theta(1)$ (simple statements)

$T_{2w}(n) = \Theta(n)$

$T_{2b}(n) = \Theta(1)$

$T_2(n) = O(n)$

inner loop => $T_{2(w)}(n) = \Theta(n)$, $T_{2(b)}(n) = \Theta(1)$] $T_2(n) = O(n)$

All loop

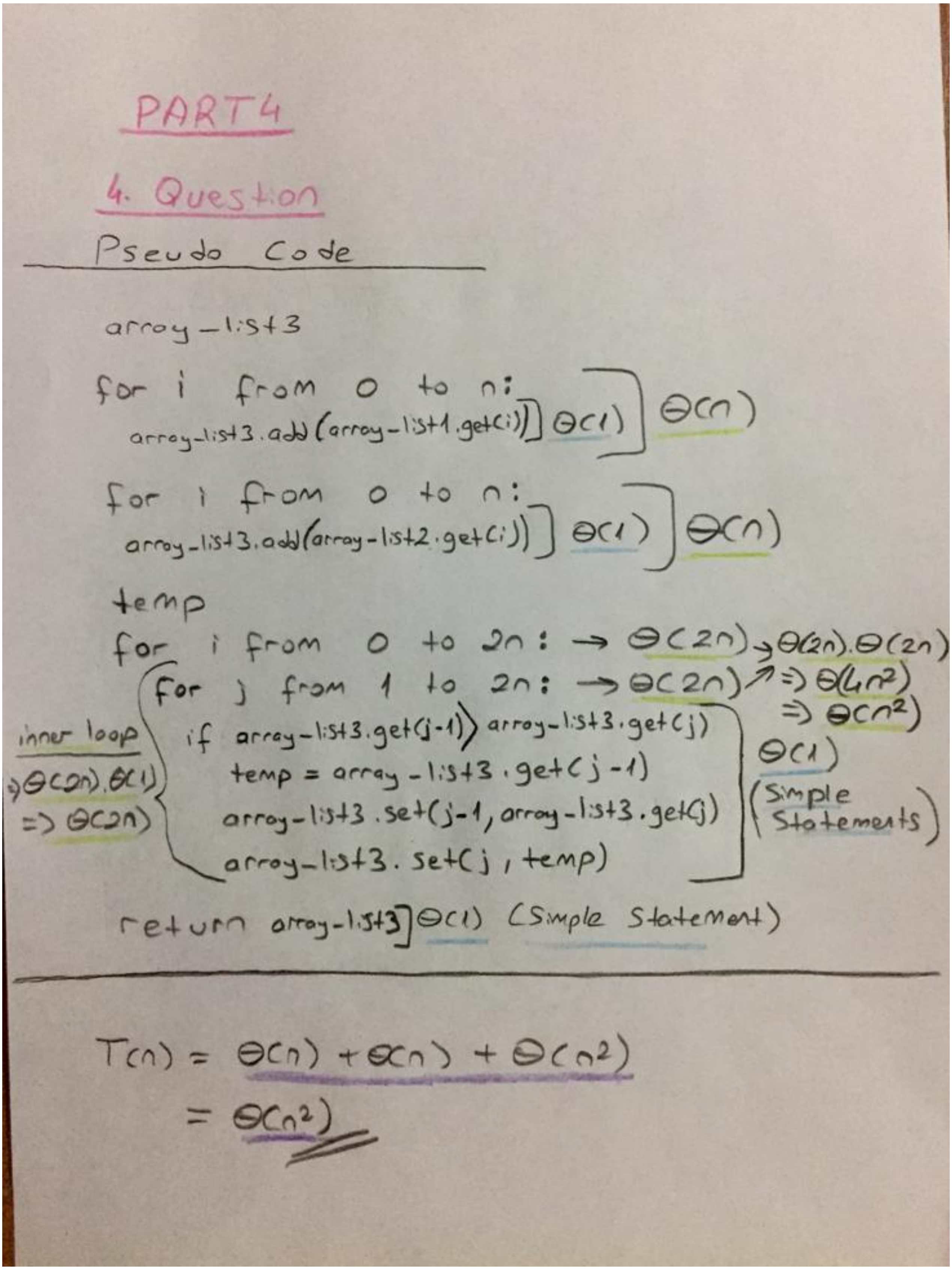
$T_{3(w)}(n) = T_{2(w)}(n) \cdot T_{1(w)}(n) = \Theta(n).\Theta(n) = \Theta(n^2)$

$T_{3(b)}(n) = T_{2(b)}(n).T_{1(b)}(n) = \Theta(1).\Theta(1) = \Theta(1)$

$T_3(n) = T_1(n) \cdot T_2(n) = O(n).O(n) = O(n^2)$

All Algorithm => $T_{(w)}(n) = \Theta(n^2)$, $T_{(b)}(n) = \Theta(1)$] $T(n) = O(n^2)$

4. Question



PART4

4. Question

Pseudo Code

```
array-list3
for i from 0 to n:
    array-list3.add(array-list1.get(i))]  $\Theta(1)$ ]  $\Theta(n)$
for i from 0 to n:
    array-list3.add(array-list2.get(i))]  $\Theta(1)$ ]  $\Theta(n)$
temp
for i from 0 to 2n: → $\Theta(2n)$ → $\Theta(2n).\Theta(2n)$
    for j from 1 to 2n: → $\Theta(2n)$ ↗ ⇒ $\Theta(4n^2)$
                                        ⇒ $\Theta(n^2)$
        if array-list3.get(j-1) > array-list3.get(j)      $\Theta(1)$
            temp = array-list3.get(j-1)                   (Simple Statements)
            array-list3.set(j-1, array-list3.get(j))
            array-list3.set(j, temp)
return array-list3]  $\Theta(1)$  (Simple Statement)
```

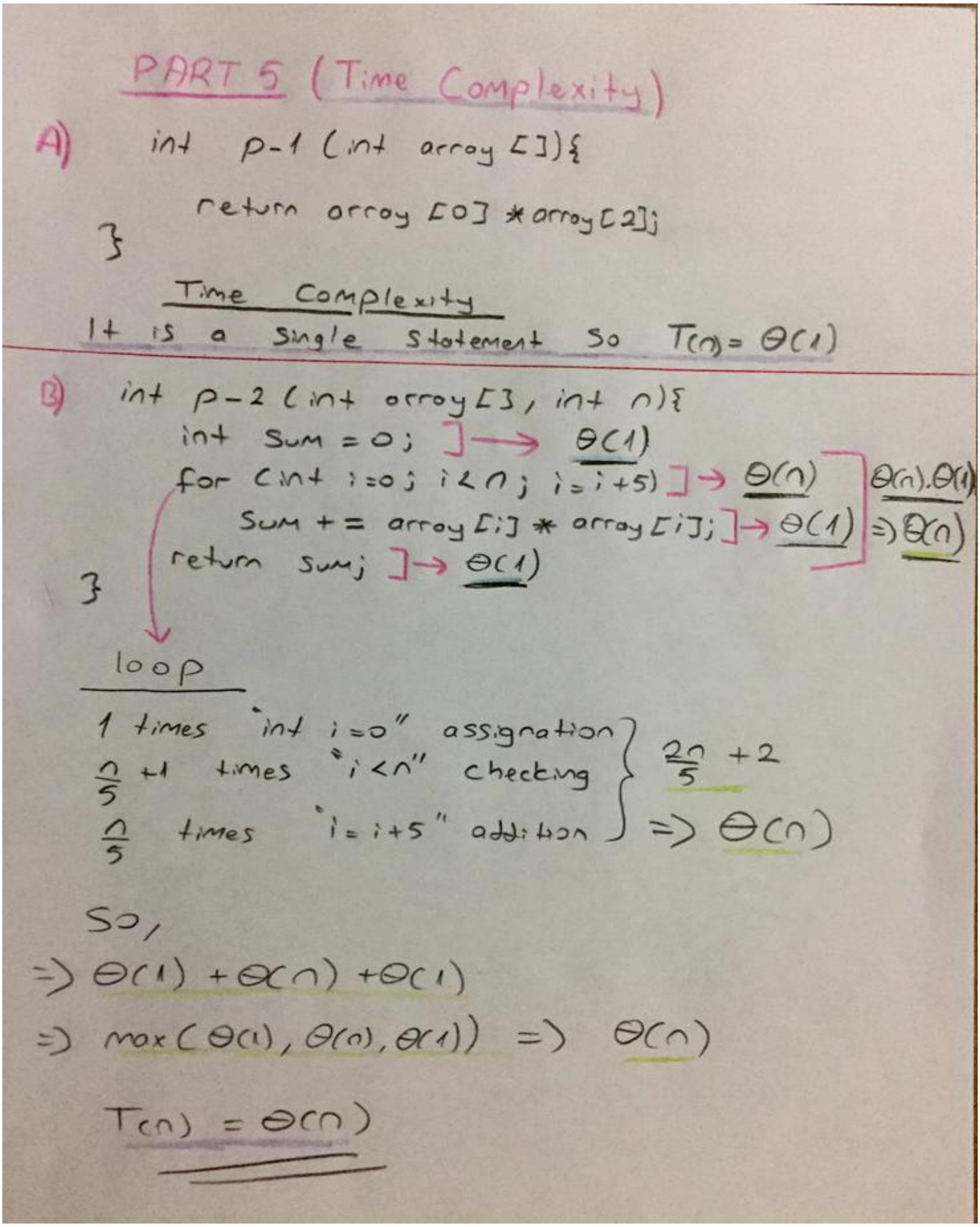
inner loop ⇒ $\Theta(2n).\Theta(1)$ ⇒ $\Theta(2n)$

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

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

PART 5

A) Ve B) Time Complexity



PART 5 (Time Complexity)

A)

```
int p-1 (int array []){
    return array [0] * array [2];
}
```

Time Complexity

It is a single statement so $T(n) = \Theta(1)$

B)

```
int p-2 (int array[], int n){
    int sum = 0; ] → Θ(1)
    for (int i=0; i<n; i=i+5) ] → Θ(n)
        sum += array[i] * array[i]; ] → Θ(1)
    return sum; ] → Θ(1)
}
```

$\Theta(n).\Theta(1) \Rightarrow \Theta(n)$

loop

1 times "int i=0" assignation

$\frac{n}{5}+1$ times "i<n" checking

$\frac{n}{5}$ times "i=i+5" addition

$\frac{2n}{5}+2 \Rightarrow \Theta(n)$

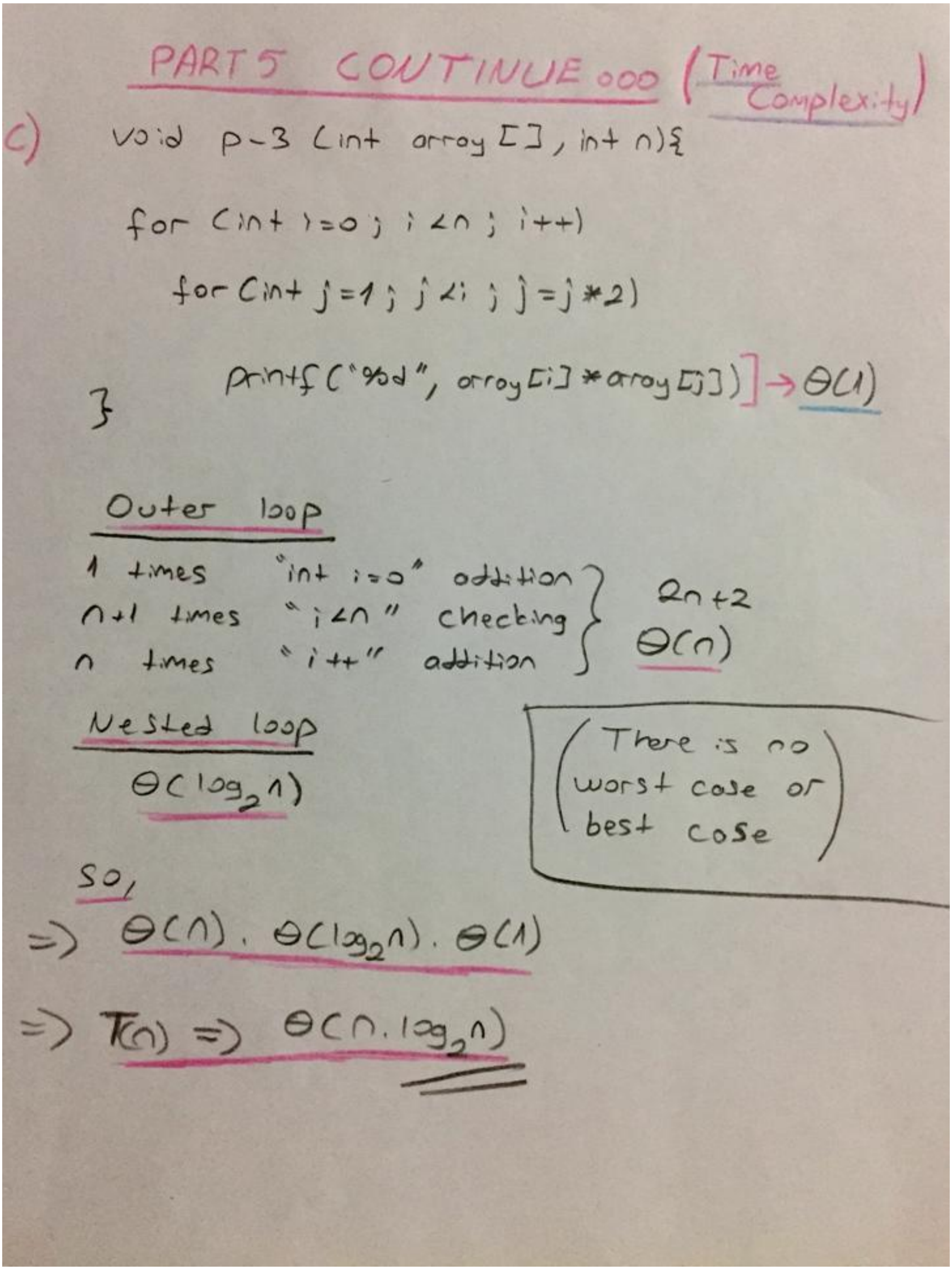
So,

$\Rightarrow \Theta(1) + \Theta(n) + \Theta(1)$

$\Rightarrow \max(\Theta(1), \Theta(n), \Theta(1)) \Rightarrow \Theta(n)$

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

C)Time Complexity



PART 5 CONTINUE ooo (Time Complexity)

c)

```
void p-3 (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])] → Θ(1)
}
```

Outer loop

1 times "int i=0" addition
n+1 times "i<n" checking
n times "i++" addition

$\}$ $2n+2$

$\Theta(n)$

Nested loop

$\Theta(\log_2 n)$

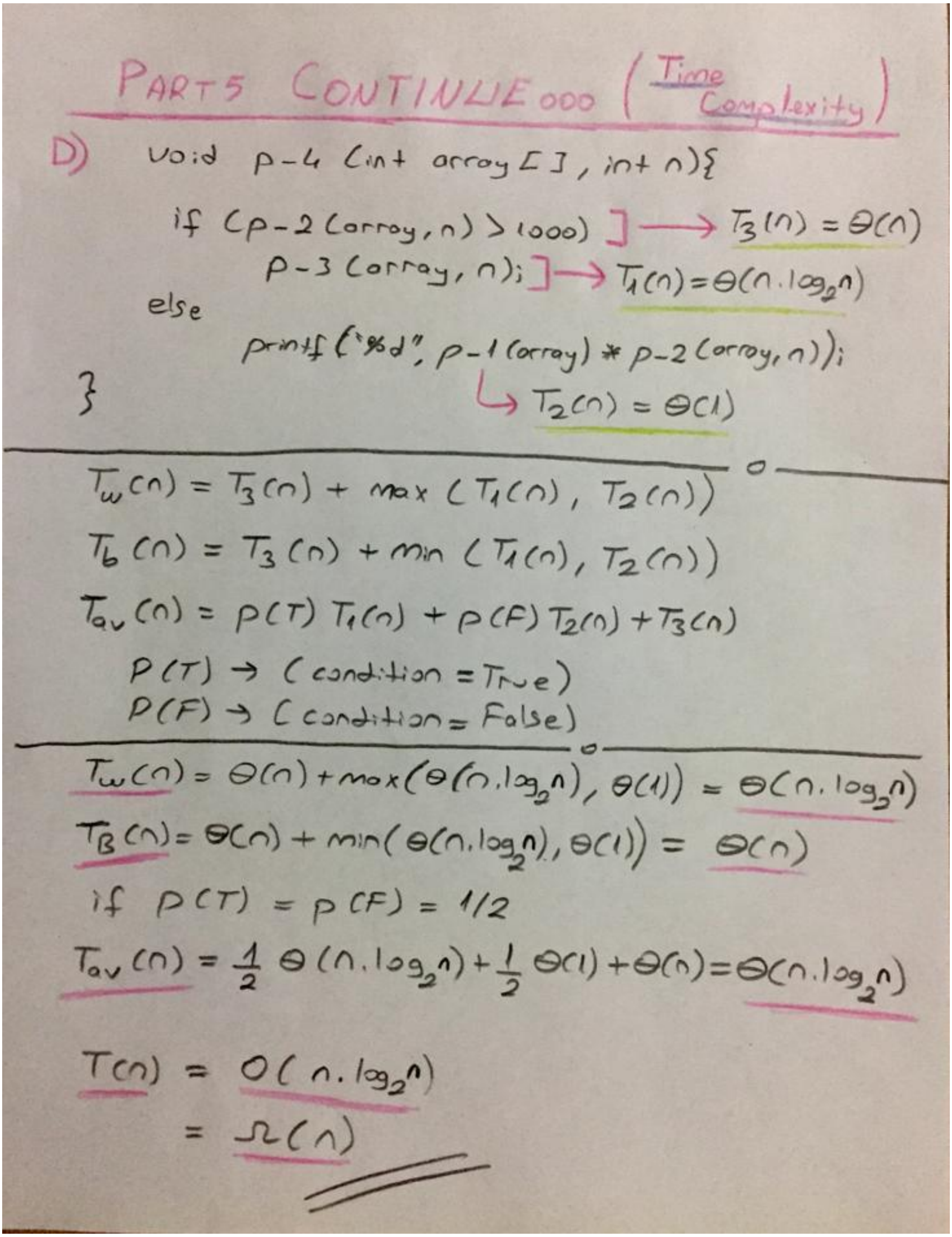
(There is no worst case or best case)

So,

$\Rightarrow \Theta(n) . \Theta(\log_2 n) . \Theta(1)$

$\Rightarrow T(n) \Rightarrow \Theta(n.\log_2 n)$

D) Time Complexity



PART5 CONTINUE ooo (Time Complexity)

D)

```
void p-4 (int array[], int n){
    if (p-2(array, n) > 1000)
        p-3(array, n);
    else
        printf("%d", p-1(array) * p-2(array, n));
}
```

$\xrightarrow{} T_3(n) = \Theta(n)$

$\xrightarrow{} T_1(n) = \Theta(n.\log_2 n)$

$\hookrightarrow T_2(n) = \Theta(1)$

$T_w(n) = T_3(n) + \max(T_1(n), T_2(n))$

$T_b(n) = T_3(n) + \min(T_1(n), T_2(n))$

$T_{av}(n) = p(T)\, T_1(n) + p(F)\, T_2(n) + T_3(n)$

$P(T) \rightarrow$ (condition = True)

$P(F) \rightarrow$ (condition = False)

$T_w(n) = \Theta(n) + \max(\Theta(n.\log_2 n), \Theta(1)) = \Theta(n.\log_2 n)$

$T_B(n) = \Theta(n) + \min(\Theta(n.\log_2 n), \Theta(1)) = \Theta(n)$

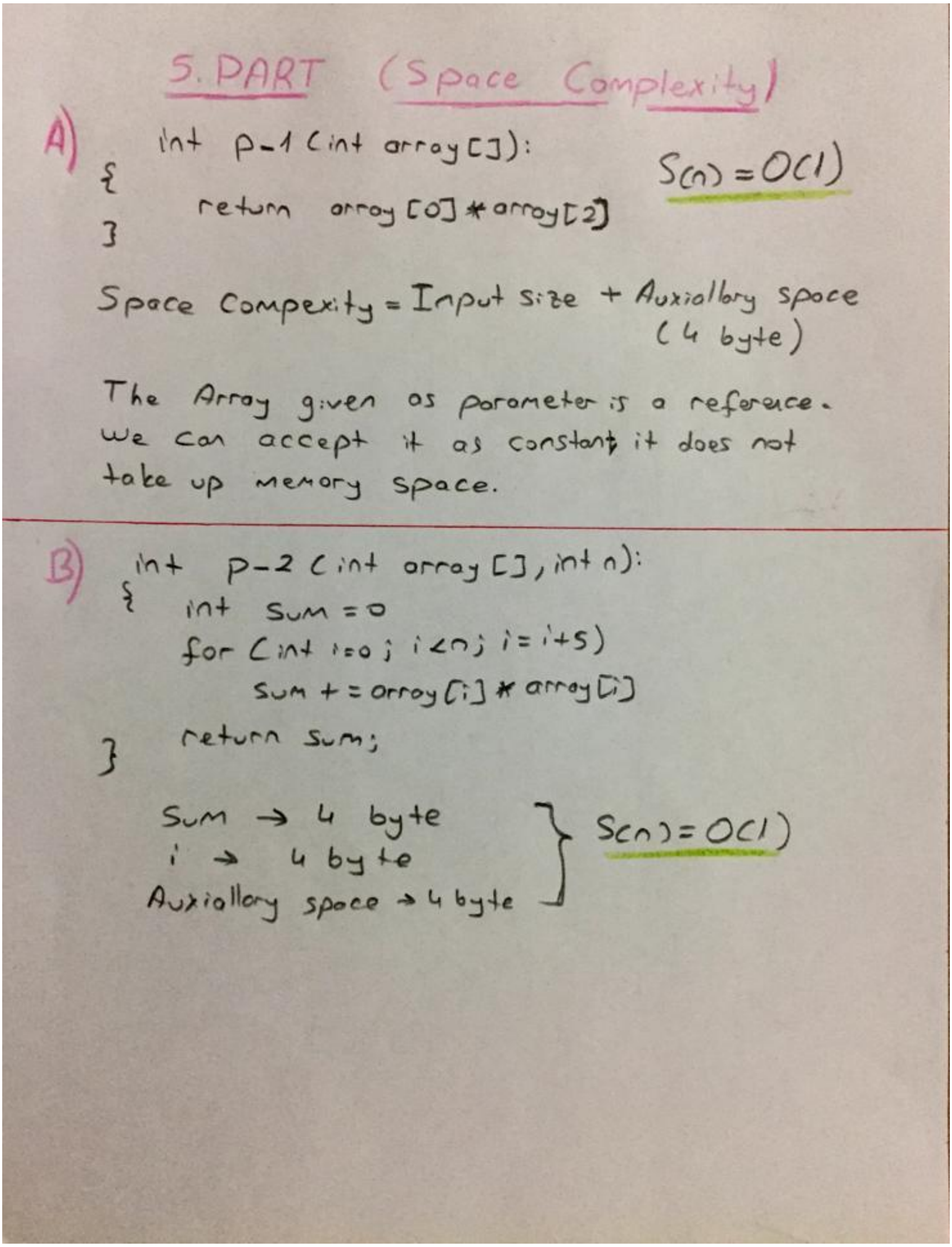
if $p(T) = p(F) = 1/2$

$T_{av}(n) = \frac{1}{2}\Theta(n.\log_2 n) + \frac{1}{2}\Theta(1) + \Theta(n) = \Theta(n.\log_2 n)$

$T(n) = O(n.\log_2 n)$

$= \Omega(n)$

A) Ve B) Space Complexity



5. PART (Space Complexity)

A)

```
int p-1 (int array[]):
{
        return array[0]*array[2]
}
```

$S(n) = O(1)$

Space Compexity = Input size + Auxiallary space
(4 byte)

The Array given as parameter is a reference. We can accept it as constant it does not take up memory space.

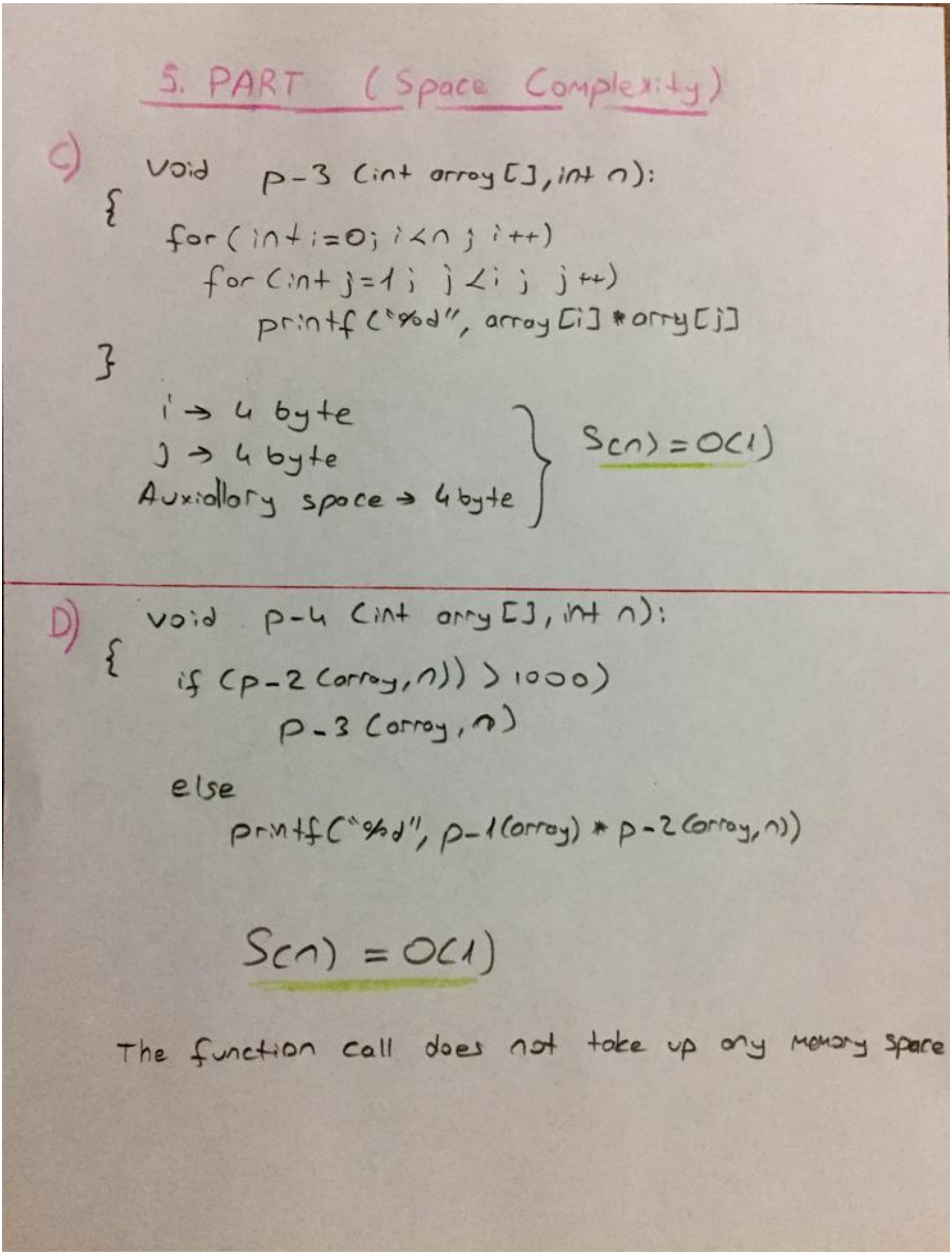
B)

```
int p-2 (int array[], int n):
{
    int sum = 0
    for (int i=0; i<n; i=i+5)
          sum += array[i]*array[i]
    return sum;
}
```

sum → 4 byte
i → 4 byte
Auxiallary space → 4 byte

$S(n) = O(1)$

B) Ve D) Space Complexity



5. PART (Space Complexity)

c)

```
void    p-3 (int arroy[], int n):
{
  for ( int i=0; i<n ; i++)
     for (int j=1 ;  j<i ;  j++)
         printf ("%d", array[i] *orry[j]
}
```

i → 4 byte
j → 4 byte
Auxiallory space → 4 byte

$S(n) = O(1)$

D)

```
void   p-4 (int orry [], int n):
{
  if (p-2 (orray,n)) > 1000)
       p-3 (orroy , n)

  else
     printf("%d", p-1(orroy) * p-2 (orroy,n))
```

$S(n) = O(1)$

The function call does not take up any memory space