PART 1
SEARCH
ADD-REMOVE

```java
public boolean product_is_in(int branch,String furniture,char model,String color){
        int model_number=(int)model;
        model_number-=65;

            if(furniture.equals("chairs")){  ⇒ O(n)
                if(color.equals("blue")){  ⇒ O(n)
                    return (katalog[branch].chairs[model_number][0]>0);  ⇒ Θ(1)
                }
                else if(color.equals("red")){
                    return (katalog[branch].chairs[model_number][1]>0);  ⇒ Θ(1)
                }
                else if(color.equals("yellow")){  ⇒ O(n)
                    return (katalog[branch].chairs[model_number][2]>0);  ⇒ Θ(1)      ⇒ O(n)
                }
                else if(color.equals("green")){  ⇒ O(n)
                    return (katalog[branch].chairs[model_number][3]>0);  ⇒ Θ(1)
                }
                else if(color.equals("red")){  ⇒ O(n)
                    return (katalog[branch].chairs[model_number][4]>0);  ⇒ Θ(1)
                }
                else if(color.equals("brown")){  ⇒ O(n)
                    return (katalog[branch].chairs[model_number][5]>0);  ⇒ Θ(1)
                }
                else
                    return false;  ⇒ Θ(1)
            }
            else if(furniture.equals("desks")){  ⇒ O(n)
                if(color.equals("blue")){  ⇒ O(n)                                  ⇒ O(n)
                    return (katalog[branch].desks[model_number][0]>0);  ⇒ Θ(1)
                }
                else if(color.equals("red")){  ⇒ O(n)
                    return (katalog[branch].desks[model_number][1]>0);  ⇒ Θ(1)
                }
                else if(color.equals("yellow")){  ⇒ O(n)
                    return (katalog[branch].desks[model_number][2]>0);  ⇒ Θ(1)      ⇒ O(n)
                }
                else if(color.equals("green")){  ⇒ O(n)
                    return (katalog[branch].desks[model_number][3]>0);  ⇒ Θ(1)
                }
                else if(color.equals("red")){  ⇒ O(n)
                    return (katalog[branch].desks[model_number][4]>0);  ⇒ Θ(1)
                }
                else
                    return false;  ⇒ Θ(1)
            }
            else if(furniture.equals("meeting_tables")){  ⇒ O(n)
                if(color.equals("blue")){  ⇒ O(n)
                    return (katalog[branch].meeting_tables[model_number][0]>0);  ⇒ Θ(1)
                }
                else if(color.equals("red")){  ⇒ O(n)                               ⇒ O(n)
                    return (katalog[branch].meeting_tables[model_number][1]>0);  ⇒ Θ(1)
                }
                else if(color.equals("yellow")){  ⇒ O(n)
                    return (katalog[branch].meeting_tables[model_number][2]>0);  ⇒ Θ(1)
                }
```

```java
public void add_product(int branch_id,String product,int color_id,int model_id )throws ahmets_exceptions{
    if(branch_id>katalog.length+1){                                              Θ(1)        Θ(1)
        throw new ahmets_exceptions("INCORRECT BRANCH ID PLEASE TRY AGAIN");     Θ(1)
    }
    if(product.equals("chairs")){                                                O(n)        O(n)
        katalog[branch_id].chairs[model_id][color_id]++;          Θ(1)
    }
    if(product.equals("desks")){                                                 O(n)        O(n)        T(n)=O(n)
        katalog[branch_id].desks[model_id][color_id]++;           Θ(1)
    }
    if(product.equals("meeting_tables")){                                        O(n)        O(n)
        katalog[branch_id].meeting_tables[model_id][color_id]++;          Θ(1)
    }
}
/**
*The desired product is added to the desired branch.just model
*@param branch_id,product,model_id
*/
public void add_product(int branch_id,String product,int model_id )throws ahmets_exceptions{
    if(branch_id>katalog.length+1){                                              Θ(1)        Θ(1)
        throw new ahmets_exceptions("INCORRECT BRANCH ID PLEASE TRY AGAIN");     Θ(1)
    }
    if(product.equals("bookcases")){                                             O(n)        O(n)        T(n)=O(n)
        katalog[branch_id].bookcases[model_id]++;          Θ(1)
    }
    if(product.equals("office_cabinets")){                                       O(n)        O(n)
        katalog[branch_id].office_cabinets[model_id]++;          Θ(1)
    }

}

/**
*The desired product is deleted to the desired branch for model and color
```

```
public void remove_product(int branch_id,String product,int color_id,int model_id )throws ahmets_exceptions{

    if(product.equals("chairs")){        O(n)
        if(katalog[branch_id].chairs[model_id][color_id]<1){        Θ(1)
            throw new ahmets_exceptions("THIS PRODUCT ALREADY NO");        Θ(1)        O(n)
        }
        katalog[branch_id].chairs[model_id][color_id]--;        Θ(1)

    }
    if(product.equals("desks")){        O(n)
        if(katalog[branch_id].desks[model_id][color_id]<1){        Θ(1)        O(n)
            throw new ahmets_exceptions("THIS PRODUCT ALREADY NO");        Θ(1)        O(n)
        }
        katalog[branch_id].desks[model_id][color_id]--;        Θ(1)
    }
    if(product.equals("meeting_tables")){        O(n)
        if(katalog[branch_id].meeting_tables[model_id][color_id]<1){        Θ(1)
            throw new ahmets_exceptions("THIS PRODUCT ALREADY NO");        Θ(1)        O(n)
        }
        katalog[branch_id].meeting_tables[model_id][color_id]--;        Θ(1)
    }

}
/**
*The desired product is deleted to the desired branch.just model
*@param branch_id,product,model_id
*/
public void remove_product(int branch_id,String product,int model_id )throws ahmets_exceptions{
    if(product.equals("bookcases")){        O(n)
        if(katalog[branch_id].bookcases[model_id]<1){        Θ(1)
            throw new ahmets_exceptions("THIS PRODUCT ALREADY NO");        Θ(1)        O(n)
        }
        katalog[branch_id].bookcases[model_id]--;        Θ(1)        O(n)
    }
    if(product.equals("office_cabinets")){        O(n)
        if(katalog[branch_id].office_cabinets[model_id]<1){        Θ(1)
            throw new ahmets_exceptions("THIS PRODUCT ALREADY NO");        Θ(1)        O(n)
        }
        katalog[branch_id].office_cabinets[model_id]--;        Θ(1)
    }

}
```

Part 2

a) Big-O is an Asymptotic notation for the word case, or ceiling of growth for a given function. It gives us an asymptotic upper bound for the growth rate of runtime of an algorithm. So we can say " at least "

b) $F(n) \leq F(n) + g(n)$ and $g(n) \leq F(n) + g(n)$

$$\max(F(n), g(n)) \in O(F(n) + g(n))$$

$F(n) + g(n) \leq 2\max(F(n), g(n))$

$$\max(F(n), g(n)) \in \Omega(F(n) + g(n))$$

we get that

$\max(F(n), g(n)) \in \Theta(F(n) + g(n))$

Note that

$$\max(F(n), g(n)) = \begin{cases} F(n) & \text{if } F(n) \geq g(n) \\ g(n) & \text{if } g(n) \geq F(n) \end{cases}$$

if $F(n) = 10n$ and $g(n) = n^2$, we get that

$$\max(F(n), g(n)) = \begin{cases} 10n & \text{if } n \leq 10 \\ n^2 & \text{if } n \geq 10 \end{cases}$$

Part 2

C) if $\lim_{n \to \infty} \frac{f(n)}{g(n)} = c \neq 0 \Rightarrow f(n) = \Theta(g(n))$

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

$\lim_{n \to \infty} \frac{2^{n+1}}{2^n} = \lim_{n \to \infty} \frac{2 . \cancel{2^n}}{\cancel{2^n}} = 2$ so true

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

$\lim_{n \to \infty} \frac{2^{2n}}{2^n} = \lim_{n \to \infty} \frac{2^n . \cancel{2^n}}{\cancel{2^n}} = \infty$ so it is false

3) $f(n) = O(n^2) \Rightarrow 0 \leq f(n) \leq cn^2$

$g(n) = \Theta(n^2) \Rightarrow c_1 n^2 \leq g(n) \leq c_2 n^2$

$\times$

$0 \leq f(n) . g(n) \leq c . c_2 . n^4$

$\Theta(n^4) = f(n) . g(n) \rightarrow c_3 . n^4 \leq f(n) . g(n) \leq c_4 . n^4$ ✗

$O(n^4) = f(n) . g(n) \rightarrow 0 \leq f(n) . g(n) \leq c_5 . n^4$ ✓

Part 3

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

exponantial $>$ n^x $>$ lineer $>$ logarithmic

- The growth rate can be compared with the limit

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

Compare $\Rightarrow$ 2^n, $n.2^n$, 3^n, 2^{n+1} $\rightarrow$ exponantial group

$$\lim_{n\to\infty} \frac{2^n}{2^{n+1}} = \lim_{n\to\infty} \frac{1}{2}\cdot\frac{\cancel{2^n}}{\cancel{2^n}} = \frac{1}{2}$$ constant so growth rate $2^n = 2^{n+1}$

$$\lim_{n\to\infty} \frac{2^n}{n.2^n} = \frac{\cancel{2^n}}{n.\cancel{2^n}} = \frac{1}{\infty} = 0$$ so growth rate $2^n.n > 2^n$

$$\lim_{n\to\infty} \frac{n.2^n}{3^n} = 0$$ so growth rate $3^n > n.2^n$

$$3^n > n.2^n > 2^{n+1} = 2^n$$

Compare $\Rightarrow$ $\log^3 n$, $\log n$ $\rightarrow$ logarithmic group

$$\lim_{n\to\infty} \frac{\log^3 n}{\log n} = \frac{\cancel{\log n}.\log^2 n}{\cancel{\log n}} = \infty$$ so growt rate $\log^3 n > \log n$

CamScanner ile tarandı

Compare $\Rightarrow n^{1.01}, \sqrt{n}, 5^{\log_2 n}, n.\log^2 n$

$$\lim_{n\to\infty} \frac{\sqrt{n}}{\cancel{n}.\log^2 n} = \lim_{n\to\infty} \lim \frac{1}{\sqrt{n}.\log^2 n} = 0$$ so growth rate $n.\log^2 n > \sqrt{n}$

$$\lim_{n\to\infty} \frac{n.\log^2 n}{n^{1.01}} = \lim \frac{\log^2 n}{n^{0.01}} = 0$$ so growth rate $n^{1.01} > \log^2 n$

$$\lim_{n\to\infty} \frac{n^{1.01}}{5^{\log_2 n}} = \lim_{n\to\infty} \frac{n^{1.01}}{n^{\log_2 5}} = 0$$ So growt rate $5^{\log_2 n} > n^{1.01}$

$5^{\log_2 n} \longrightarrow n^{\log_2 5}$

$\log_2 5 \sim 2$

$5^{\log_2 n} > n^{1.01} > n.\log^2 n > \sqrt{n}$

$$\lim_{n\to\infty} \frac{3^n}{5^{\log_2 n}} = \infty$$

$3^n > 2^n.n > 2^{n+1} = 2^n$

$$\lim_{n\to\infty} \frac{\sqrt{n}}{\log^3 n} = \infty$$

$\log n^3 > \log n$

$3^n > 2^n.n > 2^{n+1} = 2^n > 5^{\log_2 n} > n^{1.01} > n.\log^2 n > \sqrt{n} > \log_n 3 > \log n$

Part 4

1) Find the minimum-valued item

```
int P_1(int array[], int n){
    int min=array[0];  → Ø(1)
    For(int i=1; i<n; i++)  → Ø(n)
        if(min>array[i])  → Ø(1)
            min=array[i];  → Ø(1)

    return min;  → Ø(1)
}
```

} Ø(1) — Ø(1). Ø(n) = Ø(n)

$T(n) = Ø(n)$

2)
```
int P_2(int array[], int n){
    int min,tmp;  → Ø(1)
    For(int i=0; i<n-1; i++){  → Ø(n)
        min=i;  → Ø(1)
        For(int j=i+1; j<n; j++){  → Ø(n)
            if(dizi[j] < dizi[min])  } Ø(1)
                min=j;
        }
        if(min!=i){
            tmp=array[i];
            array[i]=array[min]
            array[min]=tmp;
        }
    }
    if(n%2)  → Ø(1)
        return (array[n/2] + array[n/2-1])/2;  → Ø(1)

    return array[n/2];  → Ø(1)
}
```

Ø(n). — Ø(1) — Ø(n) — Ø(n). Ø(n) = Ø(n²)

$T(n) = Ø(n^2)$

CamScanner ile tarandı

3)

Find two elements whose sum is equal to a given value

```
bool P-3(int array[], int n, int value){
    For(int i=0; i<n; i++){
        For(int j=0; j<n; j++){
            if(array[i]+array[j] == value && i!=j) -> Θ(1)
                return true; -> Θ(1)
        }
    }

    return false; -> Θ(1)
}
```

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

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

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

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

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

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

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

CamScanner ile tarandı

4)

Assume there are two ordered arraylist of n elements. Merge these two list to get a single list in increasing order

```
Void p_4 (int array1[], int array2[], int array3[], int n) {
    int i=0, j=0, k=0;  -> Ø(1)
    For(; i<n && j<n; k++) {  -> T_b(n) = Ø(n)   T_w(n) = Ø(2n) => T_b(n) = T_w(n)
                                                  = T(n) = Ø(n)
        if(array1[i] < array2[j])
            array3[k] = array1[i++];   } Ø(1)

        else
            array3[k] = array2[j];     } Ø(1)

    }
    while(i<n)  -> Ø(n)
        array3[k++] = array1[i++];  -> Ø(1)     T2(n) = Ø(n)

    while(j<n)  -> Ø(n)
        array3[k++] = array2[j++];  -> Ø(1)     T3(n) = Ø(n)
}
```

constant not important

$T(n) = T_1(n) + T_2(n) + T_3(n) = Ø(n)$

Part 5)

a)
```
int p_1(int array[])
{
    return array[0]*array[2];  → Ø(1)
}
```
$T(n) = \emptyset(1)$ $S(n) = O(1)$

b)
```
int p_2(int array[], int n)
{
    int sum = 0  → Ø(1)
    For(int i=0; i<n; i=i+5)  → Ø(n)
        sum += array[i]*array[i];  → Ø(1)
    return sum;  → Ø(1)
}
```
0, 5, 10, 15 ..... n

$5k = n$

$k = \frac{n}{5}$ $\emptyset(\frac{n}{5})$

constant

$\emptyset(n)$

$T(n) = \emptyset(n)$ $S(n) = O(1)$

c)
```
void p_3(int array[], int n)
{
    For(int i=0; i<n; i++)  → Ø(n)
        For(int j=0; j<i; j=j*2)  → Ø(logn)
            printf("%d", array[i]*array[j]);  → Ø(1)
}
```
$2^k = n \rightarrow \log n = 2$

$\emptyset(\log n)$

$\emptyset(n\log n)$

$T(n) = \emptyset(n \log n)$ $S(n) = O(n)$

d)

```
void p_4(int array[], int n)
{
        if (p_2(array,n) > 1000)        → Θ(n)        }
                p_3(array,n);           → Θ(n.logn)   }  Θ(n.logn)
        else
             printf("%d", p_1(array) * p_2(array,n))  → Θ(n)
}
```

$T(n) = \Theta(n.\log n)$ $\quad S(n) = O(n)$