PART 1:

A)

```
public boolean searchFurnitureOnBranch(Furniture tempFurniture){
      for (int i = 0; i < furniture_iterator; i++) {

            //Tb= Q(1)
                  //Tw= Q (n)
                        //T(n)=O(n)
                  if(branchFurnitures[i]==tempFurniture)//Q(1)
                              return true;
                  }
            return false;

            }
```

```
B)
public void addFurnitureByBranchName(String branchName,Furniture
tempFurniture)

{
            //T(a,m,n)=Q(m*(a+n))
            for (int i = 0; i <branch_iterator; i++){//Q(m)

      if(branches[i].getName().equals(branchName))//Q(a)+Q(n)

      branches[i].addFurniture(tempFurniture);//Q(n)
            }
}

public void addFurniture(Furniture tempFurniture){
            //T(n)=Q(n)
               branchFurnitures[furniture_iterator++]=tempFurniture;
         for (int i = 0; i < employeeIterator; i++) {//Q(n)
                 branchEmployees[i].addFurnitureToMemory(tempFurniture);
               }
}



public void addFurnitureToMemory(Furniture tempFurniture){
    branchesFurnitures[furnitureIterator++]=tempFurniture;
}//Q1
```

/*this methods best case is Q(n) because if it enters the if it should do Q(n+m*n) if it does not enters the if it should do all for loop and it is Q(n) so Q(n) is better.*/

```
public boolean sellFurniture(Furniture ordered){
    try {
           //Tw=Q(n*(n+m*n))
        //Tb=Q(n)
 //T(n)=O(n*(n+m*n))
            boolean founded=false;
              for (int i = 0; i <furniture_iterator; i++) {//Tb1=Q(1)
Tw1=Q(n+m*n)
      if(branchFurnitures[i].IsEquals(ordered)){//Q(1)
        founded=true;
      for (int j = i; j < furniture_iterator; j++) {//Q(n)
        branchFurnitures[j]=branchFurnitures[j+1];//Q(1)
       }
       furniture_iterator--;
              for (int j = 0; j < employeeIterator; j++) {//Q(m)
                   resetEmployeeProducts(branchEmployees[i]);//Q(n)
                           }
      break;
    }
        }
        if(founded==true)
 throw new MyException("There are no furnitures like you have searched");
            else
 return true;
    }catch (Exception e) {
                 System.out.println(e.getMessage());
                 return false;
    }

}



public void resetEmployeeProducts(Employee worker){
    //T(n)=Q(n)
    for (int i = 0; i <=furniture_iterator; i++){//Q(n)
        worker.branchesFurnitures[i]=branchFurnitures[i];
    }
    worker.decreaseFurnitureIterator();//Q(1)
}
```

III) my code has several functions for Querying

/* this is for admin for search all branches*/

```java
public boolean askFurniture(Furniture orderedFurniture) {

   //Tb=O(n)

//Tw=O(n)*Q(m)=O(m*n)

//Tn=O(m*n)
for (int i = 0; i < branch_iterator; i++) {//Q(m)
        if(branches[i].haveFurniture(orderedFurniture)){//O(n)
            return true;
        }
    }
    return false;
}
```

```java
//this is branch's method
public boolean haveFurniture(Furniture temp)
{
    //Tw=Q(n)
    //Tb=Q(1)
    //Tn=O(n)
    for (int i = 0; i < furniture_iterator; i++) {//Q(n)
        if(branchFurnitures[i].IsEquals(temp))//Q(1)
            return true;
    }
    return false;
}
```

```java
//This is employees searching method
public boolean askFurniture(Furniture orderedFurniture){

//Tw=Q(n)

//Tb=Q(1)

Tn=O(n)
    for (int i = 0; i < furnitureIterator; i++) {//Q(n)
        if(orderedFurniture==branchesFurnitures[i]){
            System.out.println("Yes furniture exists");
            return true;}
    }
    System.out.println("Furniture does not exists");
    return false;
}
```

PART 2:

A) **O(n^2)** means the algorithm will be **O(n^2)** slow at worst case. The word at least means that the algorithm will be **O(n^2)** slow at the best case. It can be anything less than **O(n^2)**.

Part 2

b) if big O and Ω equals Θ equals too (Θ is true)

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

$\underbrace{\max(f(n)+g(n))}_{a(n)} \leq 1 \underbrace{(f(n)+g(n))}_{b(n)}$

$a(n) = O(b(n))$

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

$f(n) \leq \max(f(n), g(n))$

$+ \; g(n) \leq \max(f(n), g(n))$

$f(n)+g(n) \leq 2 \cdot \max(f(n), g(n))$

$\frac{1}{2}(f(n)+g(n)) \leq \max(f(n), g(n))$

$a(n) = \Omega(b(n))$

p ✓

c) I) $\Theta(g(x)) = f(x)$ $\qquad 2^{n+1} = 2 \cdot 2^n$

$c_1 \cdot g(x) \leq f(x) \leq c_2 \cdot f(x)$

$\Rightarrow c_1\, 2^n \leq 2^n \cdot 2 \leq c_2 \cdot 2^n \Rightarrow c_1 \leq 2 \leq c_2$ $\qquad$ c_1 and c_2 are constant so it is true

II) $\Theta(g(x)) = f(x)$

$c_1\, g(x) \leq f(x) \leq c_2\, g(x)$

$c_1 \cdot \cancel{2^n} \leq (2^n)\cancel{(2^n)} \leq c_2 \cdot \cancel{2^n}$

$c_1 \leq 2^n \leq c_2$

$\log_2 c_1 \leq \log_2 2^n \leq \log_2 c_2$

$\log_2 c_1 \leq n \cdot \underbrace{\log_2 2}_{1} \leq \log_2 c_2 \Rightarrow \log_2 c_1 \leq n \leq \log_2 c_2$

↳ n can grows bigger than $\log_2 c_2$ so disproved

III) $O(n^2) = f(n)$ $\qquad g(n) = \Theta(n^2)$

$0 \leq f(n) \leq c n^2$ $\qquad c_1 n^2 \leq g(n) \leq c_2 n^2$

$\Theta(n^4) = f(n) \cdot g(n) \Rightarrow c_3\, n^4 \leq f(n) \cdot g(n) \leq c_4 \cdot n^4$

$0 \leq f(n) \cdot g(n) \leq c \cdot c_2\, n^4 \Rightarrow$ It comes from first equations product

but it can be 0 but $\Theta(n^4)$ can not be 0

so it is disproved

CamScanner ile tarandı

Part 3:

In this part I decide to compare all examples but for efficiency I will do it like quicksort.

Example: If we have x, y, z, t, q, y elements, I will sort it like:

if x > y, z and x is < all others; I put x in middle y and z to x's right and others to the left. Later I will choose random element from x's left and compare it with other elements (I do not have to compare it with elements where they are right of X) and when an element sets its true position I will put some mark on it.

Step 1: $n^{1.01}$

$$* \lim_{n\to\infty} \frac{n^{1.01}}{n\log^2 n} = \lim_{n\to\infty} \frac{n^{0.01}}{\log^2 n} = \lim_{n\to\infty} \frac{n^{-0.99} \times 0.01}{2\log n \cdot \frac{1}{\ln(2) n}}$$

$$= \lim_{n\to\infty} \frac{n^{-0.99}}{\frac{2\log n}{n}} = \lim_{n\to\infty} \frac{(n^{0.01})'}{(\log n)'} = \lim_{n\to\infty} \frac{n^{-0.99}}{\frac{1}{n}} = \lim n^{0.01} = \infty$$

$$\Rightarrow n^{1.01} > n\log^2 n$$

$$* \lim_{n\to\infty} \frac{n^{1.01}}{2^n} = \lim_{n\to\infty} \frac{(n^{1.01})'}{(2^n)'} = \lim_{n\to\infty} \frac{\cancel{1.01} \times n^{0.01}}{2^n \cdot \cancel{\ln 2}}$$ (ratio is unimportant when goes ∞)

$$= \lim_{n\to\infty} \frac{n^{\frac{0.01}{0.99}}}{2^n} = \lim_{n\to\infty} \frac{1}{2^n \cdot n^{0.99}} = \frac{1}{\infty} = 0$$

$$\Rightarrow n^{1.01} < 2^n$$

$$* \lim_{n\to\infty} \frac{n^{1.01}}{\sqrt{n}} = \lim_{n\to\infty} \frac{n^{1.01}}{n^{0.5}} = \lim_{n\to\infty} n^{0.51} = \infty$$

$$\Rightarrow n^{1.01} > \sqrt{n}$$

* Beside comparing $n^{1.01}$ with $\log^3 n$ compare $\log^3 n$ with $n \log^2 n$

$$\lim_{n\to\infty} \frac{\log^3(n)}{n \cdot \log^2(n)} = \lim_{n\to\infty} \frac{\log(n)}{n} \Rightarrow \text{L'Hopital} \Rightarrow \lim_{n\to\infty} \frac{\frac{1}{n}}{1} = \lim_{n\to\infty} \frac{1}{n}$$

$= 0 \Rightarrow n \cdot \log^2(n) > \log^3 n$

$n^{1.01} > n \cdot \log^2(n)$

$\Rightarrow n^{1.01} > \log^3(n)$

* $$\lim_{n\to\infty} \frac{n^{1.01}}{n \cdot 2^n} = \lim_{n\to\infty} \frac{n^{0.01}}{2^n} = \lim_{n\to\infty} \frac{n^{\frac{0.01}{0.99}}}{2^n \cdot \ln(2)} = \frac{1}{2^n \cdot n^{0.99}} = \frac{1}{\infty}$$

$= 0 \Rightarrow n^{1.01} < n \cdot 2^n$

* Beside compare $n^{1.01}$ with 3^n we can compare 2^n and 3^n if $3^n > 2^n \Rightarrow 3^n > n^{1.01}$

$$\lim_{n\to\infty} \frac{2^n}{3^n} = \lim_{n\to\infty} \left(\frac{2}{3}\right)^n = 0 \Rightarrow 2^n < 3^n$$

$\Rightarrow n^{1.01} < 3^n$

* Beside comparing $n^{1.01}$ with 2^{n+1}, we can compare 2^n and 2^{n+1}

$$\lim_{n\to\infty} \frac{2^n}{2^{n+1}} = \lim_{n\to\infty} \frac{1}{2}\left(\frac{2^n}{2^n}\right) \Rightarrow \lim_{n\to\infty} \frac{2^n}{2^{n+1}} = \frac{1}{2} \Rightarrow 2^n = 2^{n+1}$$

$\Rightarrow n^{1.01} < 2^{n+1}$

* $\lim_{n\to\infty}\left(\frac{n^{1.01}}{\log n}\right)$ = (Applying L'Hopital) $= \lim_{n\to\infty}\frac{1.01\cdot n^{0.01}}{\frac{1}{n}} = \lim_{n\to\infty} n^{1.01} = \infty$

$\Rightarrow n^{1.01} > \log n$

* exponential $> n^x >$ linear $>$ Logaritmic (I can use it at the beggining but I want to proove)

$\Rightarrow 5^{\log_2 n} > n^{1.01}$

so first step in my quicksort,

$n^{1.01} > (n\log^2 n, \sqrt{n}, \log^3(n), \log n)$

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

$n\log^2 n, \sqrt{n}, \log^3(n), \log n, \underline{n^{1.01}}, 2^n, n\cdot 2^n, 3^n, 2^{n+1}, 5^{\log_2 n}$

its place is correct now I will sort left and right

from left:

* $\lim_{n\to\infty}\frac{n\log^2(n)}{\sqrt{n}} = \lim_{n\to\infty} \sqrt{n}\log^2(n) = \infty$

$\Rightarrow \underline{n\log^2(n) > \sqrt{n}}$

* When I searching for $n^{1.01}$ I compare $n\log^2(n)$ and $\log^3(n)$ so

$\underline{n\log^2(n) > \log^3(n)}$

* $\lim_{n\to\infty}\frac{n\log^2(n)}{\log n} = \lim_{n\to\infty} n\cdot\log(n) = \infty$

$\Rightarrow \underline{n\log^2(n) > \log(n)}$

$n\log^2 n >$ all others on left so new sorting;

$\underbrace{\sqrt{n}, \log^3(n), \log(n)}, \underline{n\log^2(n)}, \underline{n^{1.01}}, 2^n, n\cdot 2^n, 3^n, 2^{n+1}, 5^{\log_2(n)}$

for these

$\lim_{n\to\infty}\frac{\log^3(n)}{\log(n)} = \lim_{n\to\infty}\log^2(n) = \infty \Rightarrow \underline{\log^3(n) > \log(n)}$

$$* \lim_{n\to\infty} \frac{\log(n)}{\sqrt{n}} = \lim_{n\to\infty} \frac{\frac{1}{n}}{\frac{1}{2\sqrt{n}}} = \lim_{n\to\infty} \frac{2\sqrt{n}}{n} = \lim_{n\to\infty} \frac{2}{\sqrt{n}} = 0$$

$$\Rightarrow \log(n) < \sqrt{n}$$

$$* \lim \frac{\log^3(n)}{\sqrt{n}} = \lim_{n\to\infty} \frac{\frac{3 \cdot \ln^2(n)}{n}}{\frac{1}{2\sqrt{n}}} = \lim_{n\to\infty} \frac{6 \cdot \ln^2(n)}{\sqrt{n}} = \lim_{n\to\infty} \frac{\frac{12\ln(n)}{n}}{\frac{1}{2\sqrt{n}}}$$

$$= \lim_{n\to\infty} \frac{24\ln(n)}{\sqrt{n}} = \lim_{n\to\infty} \frac{\frac{24}{n}}{\frac{1}{2\sqrt{n}}} = \lim_{n\to\infty} \frac{48}{\sqrt{n}} = 0$$

$$\log^3(n) < \sqrt{n}$$

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

at first I want to say $5^{\log_2 n}$ is smaller all of the elements which are right of the $n^{1.01}$ becase all others have exponential n but $\log_2(n)$ grows slower.

$$* \lim_{n\to\infty} \frac{2^n}{2^{n+1}} = \lim_{n\to\infty} \frac{1}{2} = \frac{1}{2} \Rightarrow 2^n = 2^{n+1}$$

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

$$* \lim_{n\to\infty} \frac{n \cdot 2^n}{3^n} = \lim_{n\to\infty} \left(n \cdot \left(\frac{2}{3}\right)^n\right) = \lim_{n\to\infty} \left(\frac{n}{\frac{1}{\left(\frac{2}{3}\right)^n}}\right) = \lim_{n\to\infty} \left(\frac{1}{\left(\frac{3}{2}\right)^n \ln\left(\frac{3}{2}\right)}\right)$$

$$= \lim_{n\to\infty} \left(\frac{2^n}{\ln\left(\frac{3}{2}\right) \cdot 3^n}\right) = \lim_{n\to\infty} \left(\frac{2^n \ln(2)}{\ln\left(\frac{3}{2}\right) \cdot 3^n \ln(3)}\right) = \lim_{n\to\infty} \frac{\ln^2(2) \cdot 2^x}{\ln^2(3) \ln\left(\frac{3}{2}\right) \cdot 3^x}$$

$$= \lim_{n \to \infty} \left(\frac{\ln^3(2) \cdot 2^n}{\ln^3(3) \ln\left(\frac{3}{2}\right) \cdot 3^n} \right) = \lim_{n \to \infty} \frac{\ln^4(2) \cdot 2^n}{\ln^4(3) \ln\left(\frac{3}{2}\right) \cdot 3^n} =$$

$$\lim_{n \to \infty} \frac{\ln^6(2) \cdot 2^n}{\ln^6(3) \ln\left(\frac{3}{2}\right) \cdot 3^n} = \lim_{n \to \infty} \frac{\ln^7(2) \cdot 2^n}{\ln^6(3) \ln\left(\frac{3}{2}\right) \cdot 3^n} = \lim_{n \to \infty} \frac{\ln^7(2) \cdot 2^n}{\ln^7(3) \ln\left(\frac{3}{2}\right) \cdot 3^n}$$

$$= \lim_{n \to \infty} \left(\frac{2^n}{3^n} \right) = \lim_{n \to \infty} \left(\frac{2}{3} \right)^n = 0 \Rightarrow \underline{3^n > 2^n}$$

Answer:

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

Part 4:

1:

```
int min= array[0];
for(int i=1; i<n; i++) // Θ(n)
{
    if( array[ i ]< min) //Θ(1)
        min= array[i];   // Θ(1)
}
```

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

2:

```
for(int i=0; i<n; i++) // Θ(n)
    for (int j = 0; j<n-i; j++) // Θ(n-i) => i:constant => Θ(n)
        if( array [j]>array [j+1] ) // Θ(1)
        {
            int temp = array[j]; //Θ(1)
            array[j]= array[j+1]; // Θ(1)
            array [j+1] =temp; // Θ(1)
        }
    if(n%2 ==1)   // Θ(1)
        return(array[n/2]+array[n/2 -1])/2; // Θ(1)
    return array[n/2]; // Θ(1)
/* Θ(n) * Θ(n) = Θ(n^2) */
```

3:

```
for (int i=0; i<n; i++) //T1b(n)= Θ(1) T1w(n)= Θ(n)
    for(int k= i+1; k<n; k++) //T2b(n)=Θ(1) T2w(n) = Θ(n-k)= Θ(n)
        if(array [i] + array[k] = = number) {
            print(ith and jth elements sum = number);
            retun true;}
return false;
```

$T_b(n) = \Theta_1$

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

$T_n = O(n^2)$

CamScanner ile tarandı

```
4: int i=0, j=0, k=0;
 for(;i<n && j<n; k++) //Θ(n)
   { if(arr1[i]<arr2[j])
       arr3[k]= arr1[i++];
     else
       arr3[k] = arr2[j++];
   }
   while (i<n){  //Θ(n)
       arr3[k++] = arr1[i++];}
   while(j<n){
       arr3[k++] = arr1[j++];}
 //Tn=Θ(n)
```

PART 5:

A)

int p_1 (int array[])

{

return array[0] * array[2]);// **θ** (1)

}

//Tn=θ(1),because it does not have loop.

//s(n) =O(1)

B)

int p_2 (int array[], int n)

{

int sum = 0;

for (int i = 0; i < n; i=i+5) // **θ(n/5)**

sum += array[i] * array[i]);// **θ(1)**

return sum;// **θ(1)**

}

//θ(n/5)* θ(1)= θ(n/5)= θ(n)

//Tn=Q(n)

//s(n)=O(1)

C)

void p_3 (int array[], int n)

{

for (int i = 0; i < n; i++) // **θ(n)**

for (int j = 0; j < i; j=j*2)// **θ(log(n))**

print("%d", array[i] * array[j])// **θ(1)**

}

//θ(n)* θ(log(n))* θ(1)= θ(log(n))* θ(n)= θ(n*log(n))

//s(n)=O(n) because interior for loop creates int j many times

D)

```
void p_4 (int array[], int n)
{
 If (p_2(array, n)) > 1000) // θ(n)
	p_3(array, n) // θ(n*log(n))
else
	printf("%d", p_1(array) * p_2(array, n))// θ(1)+θ(n)= θ(n)
 }



/*
We have 2 condition
If p_2>1000
θ(n)+ θ(n*log(n))= θ(n*log(n)+n)= θ(n*(log(n)+1))= θ(n*log(n))
Else we have θ(n)
So
Tw= θ(n*log(n))
Tb= θ(n)
T=O(n*log(n))
S(n)=O(n)
*/
```