```java
package Test;

interface AdvancedArithmentic{
    abstract int sumofdivisors(int n);
}
public class Problem1 implements AdvancedArithmentic{

    public int sumofdivisors(int n) {
        int sum = 0;
        for(int i=1;i*i<=n;i++) {
            if(n%i == 0) {
                if(i!=(n/i)) {
                    sum+=(i + (n/i));
                }
                else {
                    sum+=i;
                }
            }
        }
        return sum;
    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Problem1 p1 = new Problem1();
        System.out.println(p1.sumofdivisors(6));

    }

}

```

```java
package Test;
import java.io.*;
import java.util.*;
interface MenuItem{
    abstract public int pricing(int qty, int p);
}

class Sandwich implements MenuItem{

    String name;
    int price;

    Sandwich(){

    }
    Sandwich(String name, int price){
        this.name = name;
        this.price = price;
    }

    public int pricing(int qty, int price) {
        return qty*price;
    }
}

class Salad implements MenuItem{

    String name;
    int price;
    Salad(){

    }
    Salad(String name, int price){
        this.name = name;
        this.price = price;
    }

    public int pricing(int qty, int price) {
        return qty*price;
    }
}

class Drinks implements MenuItem{
    String name;
    int price;
    Drinks(){

    }
```

```
49      Drinks(String name, int price){
50          this.name = name;
51          this.price = price;
52      }
53
54      public int pricing(int qty, int price) {
55          return qty*price;
56      }
57  }
58
59  class Trio implements MenuItem{
60      public int pricing(int qty, int price) {
61          return qty*price;
62      }
63  }
64  public class Problem3 {
65
66      public static void main(String[] args) {
67          // TODO Auto-generated method stub
68          Scanner sc = new Scanner(System.in);
69          ArrayList<Sandwich> sandwich = new ArrayList<Sandwich>();
70          ArrayList<Salad> salad = new ArrayList<Salad>();
71          ArrayList<Drinks> drinks = new ArrayList<Drinks>();
72
73          sandwich.add(new Sandwich("CheeseBurger", 40));
74          sandwich.add(new Sandwich("ChickenBurger", 70));
75          sandwich.add(new Sandwich("VegBurger", 30));
76
77          salad.add(new Salad("Fruit Salad", 80));
78          salad.add(new Salad("Veg Salad", 20));
79          salad.add(new Salad("Paneer Salad", 100));
80
81          drinks.add(new Drinks("Pepsi", 60));
82          drinks.add(new Drinks("Sprite", 65));
83          drinks.add(new Drinks("Bovonto", 30));
84
85          System.out.println("Choose 1 for Trio 0 for nothing : ");
86          int choice = sc.nextInt();
87
88          if(choice == 1) {
89              String ans = "";
90              System.out.println("Enter id of each category : ");
91              System.out.println("Enter Sandwich Choice : ");
92              int sandchoice = sc.nextInt();
93              int p1 = sandwich.get(sandchoice).price;
94              ans+=sandwich.get(sandchoice).name;
95              System.out.println("Enter Salad Choice : ");
96              int saladchoice = sc.nextInt();
```

```
 97              int p2 = salad.get(saladchoice).price;
 98              ans+=salad.get(saladchoice).name;
 99              System.out.println("Enter Drinks Choice : ");
100              int drinkchoice = sc.nextInt();
101              int p3 = drinks.get(drinkchoice).price;
102              ans+=drinks.get(drinkchoice).name;
103              System.out.println("Give the number of trios : ");
104              int tot = sc.nextInt();
105              int mx = Math.max(p1,Math.max(p2, p3));
106              Trio t = new Trio();
107              System.out.println(ans);
108              System.out.println("Price is : " + t.pricing(tot, mx));
109          }
110          else if(choice == 2) {
111              System.out.println("Give the number of items : ");
112              int tot = sc.nextInt();
113              int sandchoice = sc.nextInt();
114              int p1 = sandwich.get(sandchoice).price;
115              Sandwich s1 = new Sandwich();
116              System.out.println("Price is : " + s1.pricing(tot, p1));
117          }
118          else if(choice == 3) {
119              System.out.println("Give the number of items : ");
120              int tot = sc.nextInt();
121              int sandchoice = sc.nextInt();
122              int p1 = salad.get(sandchoice).price;
123              Salad s1 = new Salad();
124              System.out.println("Price is : " + s1.pricing(tot, p1));
125          }
126          else {
127              System.out.println("Give the number of items : ");
128              int tot = sc.nextInt();
129              int sandchoice = sc.nextInt();
130              int p1 = drinks.get(sandchoice).price;
131              Drinks s1 = new Drinks();
132              System.out.println("Price is : " + s1.pricing(tot, p1));
133          }
134
135      }
136
137  }
138
```

```java

package Test;
import java.io.*;
import java.util.*;



interface digitalTree{
    abstract int absorbsunlight(int hours);

    //void getTreeDetails();
}

class BinaryTree implements digitalTree{
    public int absorbsunlight(int hours) {
        return 2*hours;
    }
}

class QuantumTree implements digitalTree{
    public int absorbsunlight(int hours) {
        return 3*hours*hours;
    }
}

class NeuralTree implements digitalTree{
    public int absorbsunlight(int hours) {
        return hours*hours*hours;
    }
}
public class ForestManager {

    String type;
    int cnt;
    int energy;

    ForestManager(){

    }

    ForestManager(String type, int cnt){
        this.type = type;
        this.cnt = cnt;
    }
    ForestManager[] forests = new ForestManager[3];

    int produceenergyforforest(int hours) {
        BinaryTree b1 = new BinaryTree();
```

```
49          QuantumTree q1 = new QuantumTree();
50          NeuralTree n1 = new NeuralTree();
51          int a = forests[0].energy = b1.absorbsunlight(hours);
52          int b = forests[1].energy = q1.absorbsunlight(hours);
53          int c = forests[2].energy = n1.absorbsunlight(hours);
54          return a*forests[0].cnt + b*forests[1].cnt + c*forests[2].cnt;
55      }
56
57      void forestReport() {
58          System.out.println("Tree"    +     "Count"       +
    "EnergyProduced");
59          int sum = 0;
60          for(int i=0;i<3;i++) {
61              sum+=forests[i].energy;
62              System.out.println(forests[i].type      +
    forests[i].cnt       +       forests[i].energy);
63          }
64          System.out.println("Total Energy Produced : " + sum);
65      }
66
67      void TestCase(){
68
69          Scanner sc = new Scanner(System.in);
70          int totalnumberoftrees = sc.nextInt();
71          //ArrayList<ForestManager> forests = new
    ArrayList<ForestManager>();
72
73          //int tree1 = 0, tree2 = 0, tree3 = 0;
74          //int[] cnt = new int[3];
75          forests[0].type = ("Binary"); forests[0].energy = 0;
76          forests[1].type = "Quantum"; forests[1].energy = 0;
77          forests[2].type = "Neem"; forests[2].energy = 0;
78          for(int i=0;i<totalnumberoftrees;i++) {
79              String type = sc.next();
80              if(type.compareTo("Binary") == 0) {
81                  forests[0].cnt++;
82              }
83              else if(type.compareTo("Quantum") == 0) {
84                  forests[1].cnt++;
85              }
86              else {
87                  forests[2].cnt++;
88              }
89          }
90      }
91      public static void main(String[] args) {
92          // TODO Auto-generated method stub
93          ForestManager fm = new ForestManager();
```

```
94          fm.TestCase();
95
96      }
97
98  }
99
```