1.

```
package kt_test2_08_08_23;

import java.util.ArrayList;
import java.util.Scanner;

interface AdvancedArithmetic
{
int divisor_sum(int n);
}

public class MyCalculator implements
AdvancedArithmetic{
@Override
public int divisor_sum(int n) {
// TODO Auto-generated method stub
ArrayList<Integer> arr = new
ArrayList<Integer>();
for(int i=1;i<=n;i++)
{
if(n%i==0)
{
arr.add(i);
}
}
```

```
int sums = summer(arr);
return sums;
}

int summer(ArrayList<Integer> arr)
{
int sums=0;
for(int i=0;i<arr.size();i++)
{
sums+=arr.get(i);

}
return sums;
}

public static void main(String args[])
{
Scanner sc = new Scanner(System.in);
int a = sc.nextInt();
MyCalculator m = new MyCalculator();
System.out.println(m.divisor_sum(a));
}


}
```

2.

```
package kt_test2_08_08_23;
```

```
import java.util.Arrays;
import java.util.Scanner;

public class SortedDictonaryValidation {

/*
 * 1st element's index in order
 * and check 2nd element contains
substring(1stelementindex:len(n))
 * if true
 * go to 2nd and check third
 */
static int Checker(String[] dict, String
orders[],String order)
{
int flag=0;
for(int i=0;i<orders.length-1;i++)
{
for(int j=i+1;j<dict.length;j++)
{
String runner[] = dict[i].split("");
int runs = order.indexOf(runner[i]);
String innerrunner[] = dict[j].split("");
```

```
if(order.substring(runs,
order.length()).contains(innerrunner[0]))
{

}
else {
flag=1;
break;
}
}
if(flag==1)
break;
}

/*

int runs=0;
for(int i=0;i<dict.length;i++)
{
String runner[] = dict[i].split("");
if(runner[i]==runner[i])
for(int j=i+1;j<dict.length;j++)
{

}
}
*/
```

```
if(flag==1)
return 0;

return 1;

}

public static void main(String args[])
{
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
int k = sc.nextInt();
String[] dict = new String[n];
for(int i=0;i<n;i++)
{
dict[i] = sc.next();
}
String order = sc.next();

String orders[] = order.split("");

System.out.println(Checker(dict,
orders,order));
}


}
```

3.

```
package kt_test2_08_08_23;

import java.util.ArrayList;
import java.util.Collections;
```

```
interface menuItem{
double menuItems_prices();
String menuItems_names();

}

class Sandwich implements menuItem
{
@Override
public double menuItems_prices() {
// TODO Auto-generated method stub

Double price = 2.75;
return price;
}
@Override
public String menuItems_names() {
// TODO Auto-generated method stub
String name = "Sandwich";
return name;
}

}
class Salad implements menuItem
{
```

```
@Override
public double menuItems_prices() {
// TODO Auto-generated method stub

Double price = 1.15;
return price;
}

@Override
public String menuItems_names() {
// TODO Auto-generated method stub
String name = "Salad";
return name;
}

}
class Drink implements menuItem
{

@Override
public double menuItems_prices() {
// TODO Auto-generated method stub

Double price = 1.25;
return price;
```

```
}

@Override
public String menuItems_names() {
// TODO Auto-generated method stub
String name = "Drink";
return name;

}

}
public class Trio implements menuItem
{
public static void main(String args[])
{
Trio t = new Trio();
System.out.println(t.menuItems_names());
System.out.println(t.menuItems_prices());
}

@Override
public double menuItems_prices() {
// TODO Auto-generated method stub
ArrayList<Double> arr = new
ArrayList<Double>();
```

```
Sandwich sd1 = new Sandwich();
Salad s1 = new Salad();
Drink d1 = new Drink();
arr.add(sd1.menuItems_prices());
arr.add(s1.menuItems_prices());
arr.add(d1.menuItems_prices());
double total_price =
(sd1.menuItems_prices()+s1.menuItems_price
s()+d1.menuItems_prices())-
(Collections.min(arr));
return total_price;
}

@Override
public String menuItems_names() {
Sandwich sd1 = new Sandwich();
Salad s1 = new Salad();
Drink d1 = new Drink();

return sd1.menuItems_names()+" /
"+s1.menuItems_names()+" / "+
d1.menuItems_names();

}

}
```

4.

```
package kt_test2_08_08_23;

import java.util.Scanner;
```

```
interface DigitalTree
{
int absorbSunlight(int hours);
String getTreeDetails();
}

class BinaryTree implements DigitalTree
{

@Override
public int absorbSunlight(int hours) {
return hours*2;
}

@Override
public String getTreeDetails() {
return "Binary Tree";
}

}
class QuantumTree implements DigitalTree
{

@Override
public int absorbSunlight(int hours) {
```

```
return (int) (3*(Math.pow(hours, 2)));
}

@Override
public String getTreeDetails() {
// TODO Auto-generated method stub
return "Quantum Tree";
}

}
class NeuralTree implements DigitalTree
{

@Override
public int absorbSunlight(int hours) {
return (int) (3*(Math.pow(hours, 3)));
}

@Override
public String getTreeDetails() {
// TODO Auto-generated method stub
return "Neural Tree";
}

}
```

```
public class ForestManager {

static int produceEnergyForForest(int
hours,int bt, int qt, int nt)
{


BinaryTree b1 = new BinaryTree();
int bt_energy = b1.absorbSunlight(hours);

QuantumTree q1 = new QuantumTree();
int qt_energy = q1.absorbSunlight(hours);

NeuralTree n1 = new NeuralTree();
int nt_energy = n1.absorbSunlight(hours);
return bt_energy*bt + qt_energy*qt +
nt_energy*nt;

}

static void getForestReport(int hours,int
bt, int qt, int nt)
{
BinaryTree b1 = new BinaryTree();
int bt_energy = b1.absorbSunlight(hours);

QuantumTree q1 = new QuantumTree();
```

```
int qt_energy = q1.absorbSunlight(hours);

NeuralTree n1 = new NeuralTree();
int nt_energy = n1.absorbSunlight(hours);

if(bt_energy>0)
{
System.out.println(b1.getTreeDetails()+"
Energy "+bt_energy*bt);
System.out.println("No of Binary Trees
"+bt);
}
if(qt_energy>0)
{
System.out.println(q1.getTreeDetails()+"
Energy "+qt_energy*qt);
System.out.println("No of Binary Trees
"+bt);
}
if(nt_energy>0)
{
System.out.println(n1.getTreeDetails()+"
Energy "+nt_energy*nt);
System.out.println("No of Binary Trees
"+nt);
}
```

```
System.out.println("Total Energy Produced
="+produceEnergyForForest(hours,bt,qt,nt))
;
}

public static void main(String args[])
{
Scanner sc = new Scanner(System.in);
System.out.println("Enter No of trees");
int a = sc.nextInt();
System.out.println("Choose Trees");
System.out.println("Enter No of Binary
trees");
int bt = sc.nextInt();
System.out.println("Enter No of Quantum
trees");
int qt = sc.nextInt();
System.out.println("Enter No of Neural
trees");
int nt = sc.nextInt();
System.out.println("Enter No of Hours");
int hours = sc.nextInt();


getForestReport(hours,bt,qt,nt);
```

```
System.out.println("Total Energy "+
produceEnergyForForest(hours,bt,qt,nt));
}
}
```