# Similar Strings Java Solution

```
import java.io.*;
import java.math.*;
import java.text.*;
import java.util.*;
import java.util.regex.*;

public class Solution {
static final int NUM_CHARS = 11;
static final int ENCODE_LENGTH = 85;

static long encode(final char[] chars, final int start, final int checkLength) {
final int length = Math.min(checkLength, chars.length-1);
long hash = 31;//5381;
int[] sim = new int[NUM_CHARS];
int count = 1;
int i=start;
for(; i <= start+length && i < chars.length; i++) {
int sim_index = chars[i] - 'a';
if(sim[sim_index] == 0) {
sim[sim_index] = count;
count++;
}
hash = hash * sim[sim_index] + 33;
}
return hash;
}

static Map<Long, List<Integer>> buildIndex(final char[] chars) {
Map<Long, List<Integer>> index = new HashMap<>();

for(int i = 0; i < chars.length - ENCODE_LENGTH; i++) {
final long encoded = encode(chars, i, ENCODE_LENGTH);
List<Integer> indexes = index.get(encoded);
if(indexes == null) {
indexes = new LinkedList<>();
index.put(encoded, indexes);
}
indexes.add(i);
}

return index;
}

static boolean isSimilar(final char[] chars, final int aStart, final int aEnd, final int bOffset) {
final int checkLength = aEnd - aStart + 1;
int[] simI = new int[NUM_CHARS+1];
int[] simJ = new int[NUM_CHARS+1];
for(int i=0; i < checkLength; i++) {
int indexI = chars[i+aStart] - 'a' + 1;
int indexJ = chars[i+bOffset] - 'a' + 1;
if(simI[indexI] == 0 && simJ[indexJ] == 0) {
simI[indexI] = indexJ;
simJ[indexJ] = indexI;
} else if(simI[indexI] != indexJ || simJ[indexJ] != indexI)
return false;
}
return true;
}
```

```
/*
* Complete the similarStrings function below.
*/
static int similarStrings(final char[] chars, int start, int end, Map<Long, List<Integer>> charIndex) {
final int sLength = chars.length;
final int checkLength = end - start + 1;
int answer = 0;
if(checkLength == 1)
answer = sLength;
else if(checkLength == ENCODE_LENGTH) {
List<Integer> indexes = charIndex.get(encode(chars,start-1, ENCODE_LENGTH));
answer = indexes == null ? 1 : indexes.size();
} else if(checkLength < ENCODE_LENGTH) {
for(int index=0; index <= sLength - checkLength; index++)
if(index == start-1 ||
isSimilar(chars, start-1, end-1, index))
answer++;
} else {
List<Integer> indexes = charIndex.get(encode(chars,start-1,ENCODE_LENGTH));
if(indexes == null)
answer = 1;
else {
for(Integer index : indexes) {
if(index + checkLength > chars.length) {
break;
} else if(index == start-1 ||
isSimilar(chars, start-1, end-1, index))
answer++;
}
}
if(answer == 0)
answer = 1;
}
return answer;
}

public static void main(String[] args) throws IOException {
final Scanner input = new Scanner(System.in);

String[] nq = input.nextLine().split(" ");
final int n = Integer.parseInt(nq[0].trim());
final int q = Integer.parseInt(nq[1].trim());

final String s = input.nextLine().trim();
final char[] sChars = s.toCharArray();
final Map<Long, List<Integer>> index = buildIndex(sChars);

StringBuilder answer = new StringBuilder(q*3);
for (int queriesRowItr = 0; queriesRowItr < q; queriesRowItr++) {
final int l = input.nextInt();
final int r = input.nextInt();

final int result = similarStrings(sChars, l, r, index);

answer.append(result);
answer.append('\n');
}
System.out.print(answer.toString());
input.close();
}
}
```

#ArrayReduction

```
package hackerrank.algorithms;

import java.util.PriorityQueue;
import java.util.Queue;

public class ArrayReduction {
static int reductionCost(int[] a) {
Queue<Integer> pq = new PriorityQueue<>();
for (int x : a)
pq.offer(x);
int cost = 0;
while (pq.size() != 1) {
int first = pq.poll();
int second = pq.poll();
cost += first + second;
pq.offer(first + second);

}
return cost;
}

public static void main(String[] args) {
int[] a = {1, 2, 3, 4};
System.out.println(reductionCost(a));
}
}
```

#Implement LRU Cache.java

```
// https://www.hackerrank.com/contests/smart-interviews/challenges/si-implement-lru-cache


import java.io.*;
import java.util.*;

public class Solution {
static class Node {
int data;
Node prev, next;
Node(int data) {
this.data = data;
this.prev = this.next = null;
}
}
public static void main(String[] args) throws NullPointerException {
Scanner sc = new Scanner(System.in);
int t = sc.nextInt();
while (t--> 0) {
int n = sc.nextInt();
int k = sc.nextInt();
int[] ar = new int[n];
for (int i = 0; i < n; i++) ar[i] = sc.nextInt();
HashMap < Integer, Node > map = new HashMap < > ();
Node dummy = new Node(-1);
Node tail = dummy;
dummy.prev = null;
int i = 0;
while (i < n) {
//hit
if (map.containsKey(ar[i])) {
if (map.get(ar[i]) != tail) {
Node curr = map.get(ar[i]);
curr.prev.next = curr.next;
curr.next.prev = curr.prev;
tail.next = curr;
curr.prev = tail;
curr.next = null;
tail = tail.next;
}
} else {
//miss and the cache is not full
if (map.size() < k) {
Node curr = new Node(ar[i]);
tail.next = curr;
curr.prev = tail;
tail = tail.next;
map.put(ar[i], curr);
}
//miss and the cache is full
else {
map.remove(dummy.next.data);
Node nn = new Node(ar[i]);
if (dummy.next.next != null) {
dummy.next = dummy.next.next;
dummy.next.prev = dummy;
tail.next = nn;
nn.prev = tail;
tail = tail.next;
map.put(ar[i], nn);
} else {
dummy.next = null;
tail = dummy;
tail.next = nn;
nn.prev = tail;
nn.next = null;
tail = nn;
map.put(ar[i], nn);
}
}
}
i++;
}

Node head = dummy.next;
Node present = head;
while (present != null) {
System.out.print(present.data + " ");
present = present.next;
}
System.out.println();
}
}
}
```

String Reduction /optimal Java Solution

```
import java.io.*;
import java.util.*;

public class Solution {

public static void main(String...args) {
Scanner sc = new Scanner(System.in);
int t = Integer.parseInt(sc.nextLine().trim());
for (int k = 0; k < t; k++) {
String s = sc.nextLine();

int[] a = new int[3];
for (int i = 0; i < s.length(); i++) {
if (s.charAt(i) == 'a') a[0]++;
if (s.charAt(i) == 'b') a[1]++;
if (s.charAt(i) == 'c') a[2]++;
}

while (true) {
int c = a[0] + a[1] + a[2];
if (a[0] == c || a[1] == c || a[2] ==c)
break;

if (a[0] <= a[1] && a[0] <= a[2]) {
a[0]++;
a[1]--;
a[2]--;
} else
if (a[1] <= a[0] && a[1] <= a[2]) {
a[1]++;
a[0]--;
a[2]--;
} else
if (a[2] <= a[0] && a[2] <= a[1]) {
a[2]++;
a[0]--;
a[1]--;
};


}

System.out.println(a[0] + a[1] + a[2]);

}
sc.close();
}
}
```

#Java Substring Comparison
// Problem: https://www.hackerrank.com/challenges/java-string-compare
// Difficulty: Easy
// Score: 10

```
import java.util.Scanner;

public class Solution {
public static String getSmallestAndLargest(String s, int k) {
String smallest = s.substring(0, k);
String largest = s.substring(0, k);
```

```
for (int i = 0; i <= s.length() - k; i++) {
String subStr = s.substring(i, k + i);
if (smallest.compareTo(subStr) > 0) {
smallest = subStr;
} else if (largest.compareTo(subStr) < 0) {
largest = subStr;
}
}
return smallest + "\n" + largest;
}

public static void main(String[] args) {
Scanner scan = new Scanner(System.in);
String s = scan.next();
int k = scan.nextInt();
scan.close();

System.out.println(getSmallestAndLargest(s, k));
}
}

#SJF_Preemptive_Scheduling.java
package rank;
import java.util.*;
public class SJF_Preemptive_Scheduling
{
public static void main(String args[])
{
Scanner sc=new Scanner(System.in);
System.out.println(" Enter The Number Of Processes - ");
int n=sc.nextInt();
int a[][]=new int[n][2];
for(int i=0;i<n;i++)
{
System.out.println(" Enter The Arrival Time & Burst Time - ");
a[i][0]=sc.nextInt();
//System.out.println(" Enter The Burst Time - ");
a[i][1]=sc.nextInt();
}
System.out.println(" PROCESS  A.T\t B.T");
for(int i=0;i<n;i++)
{
System.out.print("   P"+(i+1));
System.out.println("  \t  "+a[i][0]+"  \t "+a[i][1]);
}
Arrays.sort(a, new Comparator <int[]>(){
public int compare(final int[] e1 ,final int[] e2)
{
if(e1[0] > e2[0])
return 1;
else
return -1;
}
});
ArrayList <Integer>l=new ArrayList<Integer>();
for(int i=0;i<n;i++)
{
int k=a[i][0]+1;
a[i][i]=a[i][i]-1;
for(int j=i+1;j<n;j++)
{
if(k==a[j][0])
{
l.add(a[j][0]);
}
}
}
}
}
```

-------------------------------------------------------------

Java

#Movie-Library

```
class Film implements IFilm {
private String title;
private String director;
private int year;

@Override
public void setTitle(String title) {
this.title = title;
}

@Override
public String getTitle() {
return title;
}

@Override
public void setDirector(String director) {
this.director = director;
}

@Override
public String getDirector() {
return director;
}

@Override
public void setYear(int year) {
this.year = year;
}

@Override
public int getYear() {
return year;
}
}

class FilmLibrary implements IFilmLibrary {
private List<IFilm> films = new ArrayList<>();

@Override
public void addFilm(IFilm film) {
films.add(film);
}

@Override
public void removeFilm(String title) {
IFilm filmToRemove = null;
```

```
for (IFilm film : films) {
if (film.getTitle().equals(title)) {
filmToRemove = film;
break;
}
}
if (filmToRemove != null) {
films.remove(filmToRemove);
}
}

@Override
public List<IFilm> getFilms() {
return films;
}

@Override
public List<IFilm> searchFilms(String query) {
List<IFilm> searchResults = new ArrayList<>();
for (IFilm film : films) {
if (film.getTitle().contains(query) ||
film.getDirector().contains(query)) {
searchResults.add(film);
}
}
return searchResults;
}

@Override
public int getTotalFilmCount() {
return films.size();
}
}


#Grocery-Shop-Receipt
#item
import java.text.*;
public class Item {
public String name;
public double price;
public int bulkQuantity;
public double bulkPrice;

public Item(String name, double price) {
this.name = name;
this.price = price;
if (price < 0.0) {
throw new IllegalArgumentException();
}
}

public Item(String name, double price, int bulkQuantity, double bulkPrice) {
this.name = name;
this.price = price;
this.bulkQuantity = bulkQuantity;
this.bulkPrice = bulkPrice;
if (price < 0.0 || bulkQuantity < 0 || bulkPrice < 0.0) {
throw new IllegalArgumentException();
}
```

```
}

public double priceFor(int quantity) {
if (quantity < 0) {
throw new IllegalArgumentException();
}
if (bulkQuantity == 0) {
return price * quantity;
}
int remainder = quantity % bulkQuantity;
int finalBulk = (int) Math.floor(quantity / bulkQuantity);
return price * remainder + finalBulk * bulkPrice;
}

public String toString() {
NumberFormat nf = NumberFormat.getCurrencyInstance();
String priceText = nf.format(price);
String bulkPriceText = nf.format(bulkPrice);

if (bulkQuantity > 0) {
String newLine = String.format("    ");
return String.format(this.name + newLine + priceText + newLine + bulkQuantity + "/" + bulkPriceText);
} else {
String newLine = String.format("    ");
return String.format(this.name + newLine + priceText + newLine);
}
}
}

#Item Order
import java.text.NumberFormat;
public class ItemOrder {
public Item item;
public int quantity;

public ItemOrder(Item item, int quantity) {
this.item = item;
this.quantity = quantity;
}

public double getPrice() {
return item.priceFor(quantity);
}

public Item getItem() {
return item;
}

public String toString() {
return item.name + " " + quantity + " " + (NumberFormat.getCurrencyInstance().format(getPrice()));
}
}

#Shoping_Cart
import java.util.ArrayList;

public class ShoppingCart {
private ArrayList<ItemOrder> itemOrders;
private boolean isDiscount;
```

```
public ShoppingCart() {
itemOrders = new ArrayList<ItemOrder>();
isDiscount = false;
}

public void add(ItemOrder itemOrder) {
ItemOrder duplicate = null;
for (ItemOrder io : itemOrders) {
if (io.item.name.equals(itemOrder.item.name)) {
duplicate = io;
}
}
if (duplicate != null) {
itemOrders.remove(duplicate);
}
itemOrders.add(itemOrder);
}

public void setDiscount(boolean isDiscount) {
this.isDiscount = isDiscount;
}

public double getTotal() {
double total = 0.0;
for (ItemOrder io : itemOrders) {
total = total + io.getPrice();
}
if (isDiscount) {
total = total * 0.9;
}
return total;
}

public String getMessage() {
StringBuilder items = new StringBuilder();
for (ItemOrder io : itemOrders) {
if (io.quantity != 0) {
items.append(io.toString() + "\r\n");
}
}
return items.toString();
}

public String getDiscount() {
StringBuilder discount = new StringBuilder();
String temp = isDiscount ? "Yes" : "No";
discount.append("10% Membership Discount: " + temp + "\n");
return discount.toString();
}

public void cancel() {
itemOrders.clear();
isDiscount = false;
}
}

#ShopingMain
import java.io.File;
import java.io.FileNotFoundException;
import java.util.List;
```

```
public class ShoppingMain {
public static void main(String[] args) throws FileNotFoundException {
String fileName = "grocery.txt";
List<Item> prodList = CatalogReader.read(fileName);
Catalog list = new Catalog("Safeway Groceries");
for (Item item : prodList) {
list.add(item);
}

RegisterFrame f = new RegisterFrame(list);
f.setVisible(true);
}
}
```

Hackerrank-nutrition-chain

```
abstract class Food {
double proteins;
double fats;
double carbs;
double tastyScore;

public abstract void getMacroNutrients();
}

class Egg extends Food {
public Egg(double proteins, double fats, double carbs) {
this.proteins = proteins;
this.fats = fats;
this.carbs = carbs;
}

int tastyScore = 7;
String type = "non-vegetarian";

@Override
public void getMacroNutrients() {
System.out.println("An egg has " + this.proteins + " gms of protein, " + this.fats +
" gms of fats and " + this.carbs + " gms of carbohydrates.");
}
}

class Bread extends Food {
public Bread(double proteins, double fats, double carbs) {
this.proteins = proteins;
this.fats = fats;
this.carbs = carbs;
}

int tastyScore = 8;
String type = "vegetarian";

@Override
public void getMacroNutrients() {
System.out.println("A slice of bread has " + this.proteins + " gms of protein, "
+ this.fats + " gms of fats and " + this.carbs + " gms of carbohydrates.");
}
}
```

```
------------------------------

import java.util.Scanner;

abstract class Food {
double proteins;
double fats;
double carbs;
double tastyScore;

abstract void getMacroNutrients();
}

class Egg extends Food {
int tastyScore = 7;
String type = "non-vegetarian";

Egg(double proteins, double fats, double carbs) {
this.proteins = proteins;
this.fats = fats;
this.carbs = carbs;
}

public void getMacroNutrients() {
System.out.println("An egg has " + this.proteins + " gms of protein, " + this.fats +
" gms of fats and " + this.carbs + " gms of carbohydrates.");
}
}
-------------------------------------------------------------------

class Bread extends Food {
int tastyScore = 8;
String type = "vegetarian";

Bread(double proteins, double fats, double carbs) {
this.proteins = proteins;
this.fats = fats;
this.carbs = carbs;
}

public void getMacroNutrients() {
System.out.println("A slice of bread has " + this.proteins + " gms of protein, "
+ this.fats + " gms of fats and " + this.carbs + " gms of carbohydrates.");
}
}

public class Solution {
public static void main(String args[]) throws Exception {
Scanner sc = new Scanner(System.in);
int cnt = Integer.parseInt(sc.nextLine());

for (int i = 0; i < cnt; i++) {
String name = sc.nextLine();

if (name.equals("Bread")) {
Bread breadObj = new Bread(4, 1.1, 13.8);
for (int j = 0; j < 3; j++) {
String command = sc.nextLine();
if (command.equals("getMacros")) {
```

```
breadObj.getMacroNutrients();
} else if (command.equals("getTaste")) {
System.out.println("Taste: " + breadObj.tastyScore);
} else if (command.equals("getType")) {
System.out.println("Bread is " + breadObj.type);
}
}
} else if (name.equals("Egg")) {
Egg eggObj = new Egg(6.3, 5.3, 0.6);
for (int j = 0; j < 3; j++) {
String command = sc.nextLine();
if (command.equals("getMacros")) {
eggObj.getMacroNutrients();
} else if (command.equals("getTaste")) {
System.out.println("Taste: " + eggObj.tastyScore);
} else if (command.equals("getType")) {
System.out.println("Egg is " + eggObj.type);
}
}
}
}
}
}
```

--------------------------------

#Zoo Management

```
class Animal implements IAnimal {
private int id;
private String species;
private String name;
private int age;

@Override
public void setId(int id) {
this.id = id;
}

@Override
public int getId() {
return id;
}

@Override
public void setSpecies(String species) {
this.species = species;
}

@Override
public String getSpecies() {
return species;
}

@Override
public void setName(String name) {
this.name = name;
}

@Override
public String getName() {
return name;
```

```
}

@Override
public void setAge(int age) {
this.age = age;
}

@Override
public int getAge() {
return age;
}
}

class Zoo implements IZoo {
private List<IAnimal> animals = new ArrayList<>();

@Override
public void addAnimal(IAnimal animal) {
animals.add(animal);
}

@Override
public void removeAnimal(int id) {
Iterator<IAnimal> iterator = animals.iterator();
while (iterator.hasNext()) {
IAnimal animal = iterator.next();
if (animal.getId() == id) {
iterator.remove();
}
}
}

@Override
public int countAnimals() {
return animals.size();
}

@Override
public List<IAnimal> getAnimalsBySpecies(String species) {
List<IAnimal> specAnimals = new ArrayList<>();
for (IAnimal animal : animals) {
if (animal.getSpecies().equals(species)) {
specAnimals.add(animal);
}
}
return specAnimals;
}

@Override
public List<Map.Entry<Integer, List<IAnimal>>> getAnimalsByAge() {
Map<Integer, List<IAnimal>> ageGroups = new HashMap<>();
for (IAnimal animal : animals) {
int age = animal.getAge();
if (!ageGroups.containsKey(age)) {
ageGroups.put(age, new ArrayList<IAnimal>());
}
ageGroups.get(age).add(animal);
}

List<Map.Entry<Integer, List<IAnimal>>> result = new ArrayList<>(ageGroups.entrySet());
```

```
Collections.sort(result, new Comparator<Map.Entry<Integer, List<IAnimal>>>() {
@Override
public int compare(Map.Entry<Integer, List<IAnimal>> o1, Map.Entry<Integer, List<IAnimal>> o2) {
return o2.getKey().compareTo(o1.getKey());
}
});

return result;
}
}
```
---------------------------------------
#Compare User List
```
class UserManager {
public static List<User>[] compareUsers(List<User> usersListInDB, List<User> newUsersList) {
List<User> updated = new ArrayList<User>();
List<User> inserted = new ArrayList<User>();

for (User user : newUsersList) {
User userInDB = null;
for (User u : usersListInDB) {
if (u.getId() == user.getId()) {
userInDB = u;
break;
}
}

if (userInDB != null) {
for (java.lang.reflect.Field field : User.class.getDeclaredFields()) {
field.setAccessible(true);
try {
Object valueInDB = field.get(userInDB);
Object valueInNewList = field.get(user);

if ((valueInDB == null && valueInNewList == null) || (valueInDB == null || valueInNewList == null)) {
continue;
}

if (!valueInDB.equals(valueInNewList)) {
updated.add(user);
break;
}
} catch (IllegalAccessException e) {
e.printStackTrace();
}
}
} else {
inserted.add(user);
}
}
List<User>[] res = new ArrayList[2];
res[0] = updated;
res[1] = inserted;
return res;
}
}
```

--------------------------------------------

#Hackerrank-car-building.java

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

class Car{

// fields
int no_of_tires = 4;
String bodyType = "Plastic";

// constructor is using built in (invisible)



// method
public void reverseGear(){
System.out.println("Reverse Gear is Applied...");
}

// method
public void switchOnHeadlights(){
System.out.println("Headlights turned on...");
}
}

class BMW extends Car{

// fields
String modelName = "X3";


// method
public void topSpeed(){
System.out.println("TopSpeed of BMW is 200 kmph");
}
}

public class Solution {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
for (int i = 0; i < 3; i++) {
String name = sc.nextLine();
BMW b = new BMW();
if(name.equals("reverseGear")){
b.reverseGear();
}
if(name.equals("switchOnHeadlights")){
b.switchOnHeadlights();
}
if(name.equals("topSpeed")){
b.topSpeed();
}
}
}
}
```

!!!!!!!!!!!!!! #20 Car Fueling

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

class Car{

// An example of overriding;

public void topSpeed(){
System.out.println("Top Speed of the vehicle is 100 kmph");
}

public void fuelType(){
System.out.println("Car fuel type is Petrol");
}
}

class SUV extends Car{

@Override
public void fuelType(){
System.out.println("SUV fuel type is Diesel");
}
}

public class Solution{
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
for(int i=0;i<2;i++) {
String input = sc.nextLine();
Car suv = new SUV();
if(input.equals("topSpeed")){
suv.topSpeed();
}
if(input.equals("fuelType")){
suv.fuelType();
}
Car car = new Car();
if(input.equals("topSpeed")){
car.topSpeed();
}
if(input.equals("fuelType")){
car.fuelType();
}
}
}
}
```

!!!!!!!!!!! 21. Car Engine

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;



class Car{

// example of method overloading

public void printTopSpeed(){
System.out.println("Top speed of the vehicle is 100 kmph");
}

public void printTopSpeed(int topSpeed){
System.out.println("Top speed of the vehicle is " + topS
peed + " kmph");
}

public void printTopSpeed(String vehicleName, int topSpeed){
System.out.println("Top speed of the vehicle " + vehicle
ame + " is " + topS
peed + " kmph");
}
}
public class Solution {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
String sub = sc.nextLine();
int n = Integer.parseInt(sub);
for(int i=0;i<n;i++) {
String[] input = sc.nextLine().split(" ");
Car c = new Car();
if(input.length ==1){
c.printTopSpeed();
}
if(input.length ==2){
c.printTopSpeed(Integer.parseInt(input[1]));
}
if(input.length ==3){
c.printTopSpeed(input[1], Integer.parseInt(input[2]));
}
}
}
}
```
#Hackerrank-employee-profile.java

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

abstract class Employee {

abstract void setSalary(int salary);

abstract int getSalary();

abstract void setGrade(String grade);
abstract String getGrade();

void label(){
System.out.println("Employee's data:");
}
}

class Engineer extends Employee {
private int salary;
private String grade;

void setSalary(int salary){
this.salary = salary;
}

public int getSalary(){
return this.salary;
}

void setGrade(String grade){
this.grade = grade;
}

public String getGrade(){
return this.grade;
}
}
class Manager extends Employee {
private int salary;
private String grade;

void setSalary(int salary){
this.salary = salary;
}

public int getSalary(){
return this.salary;
}

void setGrade(String grade){
this.grade = grade;
}
public String getGrade(){
return this.grade;
}
}
public class Solution {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
String sub = sc.nextLine();
int n = Integer.parseInt(sub);
for(int i=0;i<n;i++){
String[] input = sc.nextLine().split(" ");
if(input[0].equals("ENGINEER")){
Engineer e = new Engineer();
e.setSalary(Integer.parseInt(input[2]));
e.setGrade(input[1]);
e.label();
System.out.println("GRADE : " + e.getGrade());
System.out.println("SALARY : " + e.getSalary());
}
if(input[0].equals("MANAGER")){
Manager e = new Manager();
e.setSalary(Integer.parseInt(input[2]));
e.setGrade(input[1]);
e.label();
System.out.println("GRADE : " + e.getGrade());
System.out.println("SALARY : " + e.getSalary());
}
}
}
}
```

#Java Point and Line

```
class LineList implements ListOfLines {
Vector<Line> lines;

public LineList(Vector<Line> lines) {
this.lines = lines;
}

@Override
public Vector<Line> getLinesFromStartingPoint(Point p) {
Vector<Line> res = new Vector<>();
for (int i = 0; i < lines.size(); i++) {
Line curlin = lines.elementAt(i);
if (test(curlin.getStart(), p))
res.add(curlin);
}
return res;
}

private boolean test(Point a, Point b) {
return a.getX() == b.getX() && a.getY() == b.getY();
}

public Line getLineWithMaxLength() {
Line maxl = lines.elementAt(0);
for (int i = 1; i < lines.size(); i++) {
if (lines.elementAt(i).getLineLength() > maxl.getLineLength())
maxl = lines.elementAt(i);
}
return maxl;
}
}
```

-----------------------------------

#Bank ChatBot

```
class BankOperations implements IBankAccountOperation {
private double balance;

public void deposit(double d) {
balance += d;
}

public double processOperation(String message) {
String[] words = message.split(" ");
double value = 0;
for (String word : words) {
try {
value = Double.parseDouble(word);
break;
} catch (NumberFormatException e) {
//do nothing
}
}
if (message.toLowerCase().contains("deposit") || mess
age.toLower
Case().contains("invest")
|| message.toLowerCase().contains("transfer")) {
deposit(value);
} else if (message.toLowerCase().contains("withdraw") || message.toLo
wer
Case().contains("pull")) {
withdraw(value);
}
return balance;
}

public void withdraw(double d) {
if (balance >= d) {
balance -= d;
}
}
}
```

-----------------------------------------

#Help-Dest Ticket Processing

```
class Employee implements IEmployee {
private String fullName;
private int pointLevel;
private List <Category> assignedCategories;

public Employee(String fullName, int pointLevel, List <Category>
assignedCategories) {
this.fullName = fullName;
this.pointLevel = pointLevel;
this.assignedCategories = assignedCategories;
}

public void setFullName(String fullName) {
this.fullName = fullName;
}

public String getFullName() {
return this.fullName;
}

public void setPointLevel(int pointLevel) {
this.pointLevel = pointLevel;
}

public int getPointLevel() {
return this.pointLevel;
}

public void setAssignedCategories(List <Category>
assignedCategories) {
this.assignedCategories = assignedCategories;
}
```

```
public List <Category> getAssignedCategories() {
return this.assignedCategories;
}
}

class Ticket implements ITicket {
private int id;
private String name;
private Category category;
private int point;
private boolean isCompleted;
private String assignedEmployee;

public Ticket(int id, String name, Category category,
int point) {
this.id = id;
this.name = name;
this.category = category;
this.point = point;
this.isCompleted = false;
}

public void setId(int id) {
this.id = id;
}

public int getId() {
return this.id;
}

public void setName(String name) {
this.name = name;
}

public String getName() {
return this.name;
}

public void setCategory(Category category) {
this.category = category;
}

public Category getCategory() {
return this.category;
}

public void setPoint(int point) {
this.point = point;
}

public int getPoint() {
return this.point;
}

public void setIsCompleted(boolean isCompleted) {
this.isCompleted = isCompleted;
}

public boolean getIsCompleted() {
return this.isCompleted;
```

```
}

public void setAssignedEmployee(String assignedEmployee) {
this.assignedEmployee = assignedEmployee;
}

public String getAssignedEmployee() {
return this.assignedEmployee;
}
}

class HelpDesk implements IHelpDesk {
private List<IEmployee> employees;
private List<ITicket> tickets;

public HelpDesk() {
employees = new ArrayList<IEmployee>();
tickets = new ArrayList<ITicket>();
}

public void addTicket(ITicket ticket) {
tickets.add(ticket);
}

public void addEmployee(IEmployee employee) {
employees.add(employee);
}

public void completeTicket(String employeeFullName, int ticketId) {
IEmployee employee = employees.stream().filter(e -> e.getFullName().equals
(employeeFullName)).findFirst().orElse(null);
if (employee == null) {
return;
}
ITicket ticket = tickets.stream().filter(t -> t.getId() == tic
ketId && !t.getIsCompleted()
&& t.getPoint() <= employee.getPointLevel()).findFirst().orElse(null);
if (ticket != null && employee.getAssignedCategories().contains(tic
ket.getCategory())
&& employee.getPointLevel() >= ticket.getPoint()) {
ticket.setIsCompleted(true);
ticket.setAssignedEmployee(employeeFullName);
}
}

public int getWaitingTicketCount() {
return (int) tickets.stream().filter(t -> !t.getIsCompleted()).count();
}

public int getCompletedTicketsTotalPoint() {
return tickets.stream().filter(ITicket::getIsCompleted).mapToInt(ITi
cket::getPoint).sum();
}

public List<CategoryNode> getTicketsTotalPointByCategory() {
List<CategoryNode> result = new ArrayList<>();
for (Category category : EnumSet.allOf(Category.class)) {
int totalPoint = tickets.stream().filter(t -> t.getCategory() == cate
gory).mapToInt
(ITicket::getPoint).sum();
result.add(new CategoryNode(category, totalPoint));
}
return result;
}

public List<EmployeeNode> getTicketsTotalPointByEmployee() {
List<EmployeeNode> result = new ArrayList<>();
for (IEmployee employee : employees) {
int totalPoint = tickets.stream().filter(t -> t.getIsCompleted() && t.getAs
signedEmployee().equals(employee.getFullName())).mapToInt(ITicket::getP
oint).sum();
result.add(new EmployeeNode(employee, totalPoint));
}
return result;
}
}
--------------------------------------
#Logistic space Calculation
class SolidProduct extends Product {
private int Weight;
private int Volume;
public SolidProduct(int weight, int volume) {
super(0, 3);
Weight = weight;
Volume = volume;
}
public int CalculateSpace() {
return Weight * Volume * Factor;
}
}

class LiquidProduct extends Product {
private int Liter;
public LiquidProduct(int liter) {
super(0, 2);
Liter = liter;
}
public int CalculateSpace() {
return Liter * Factor;
}
}

class JewelProduct extends Product {
private int Count;
private int RequiredBox;
public JewelProduct(int count, int requiredBox) {
super(0, 1);
Count = count;
RequiredBox = requiredBox;
}
public int CalculateSpace() {
return Count * Factor + RequiredBox * Factor;
}
}

class TransportUnit implements ITransportUnit {
private List<Product> products = new ArrayList<Product>();
public void AddProduct(Product product) {
products.add(product);
}
```

```
public int GetTotalSpace() {
int totalSpace = 0;
for (Product item : products) {
totalSpace += item.CalculateSpace();
}
return totalSpace;
}
}
----------------------------------------
#Andor Sequences
public static Return findSequence(long x, long y) {
long mod = (long)1e9 + 7;
long count = 1;
ArrayList <Long> ret = new ArrayList <>();
int bitx = bitcount(x);
int bity = bitcount(y);
for (int i = bitx; i <= bity; i++) {
long low = Math.max(1l << (i - 1), x);
long high = Math.min((1l << i) - 1, y);
count = (count * ((high - low + 1) % mod)) % mod;
ret.add(high);
}
Return ans = new Return((int)count, ret);
return ans;
}
public static int bitcount(long x) {
int count = 0;
while (x > 0) {
count++;
x /= 2;
}
return count;
}
```