**TYPE 1**

**Alphabet Game 1**

public class JavaAssignment {

public static void main(String[] args) {

java.util.Scanner kb = new java.util.Scanner(System.in);

StringBuffer sb = new StringBuffer(kb.next());

while(sb.length()!=1) {

String s = kb.next();

System.out.println("char = "+s);

int index;

char c = s.charAt(0);

for(char i='a';i<c;i++) {

if((index=sb.indexOf(String.valueOf(i))) != -1)

sb.deleteCharAt(index);

}

System.out.println(sb);

}

kb.close();

}

}

**Alphabet Game 2**

public class JavaAssignment {

public static void main(String[] args) {

java.util.Scanner kb = new java.util.Scanner(System.in);

StringBuffer sb = new StringBuffer(kb.next());

while(sb.length()!=1) {

String s = kb.next();

System.out.println("char = "+s);

int index;

char c = s.charAt(0);

for(char i='z';i>c;i--) {

if((index=sb.indexOf(String.valueOf(i))) != -1)

sb.deleteCharAt(index);

}

System.out.println(sb);

}

kb.close();

}

}

**Alphabet Game 3**

import java.util.Collections;

import java.util.LinkedList;

import java.util.Scanner;

public class JavaAssignment {

public static void main(String[] args) {

java.util.Scanner kb = new java.util.Scanner(System.in);

StringBuffer sb = new StringBuffer(kb.next());

int turn = kb.nextInt();

kb.close();

int[] arrOfCount = new int[26];

for(int j=0;j<sb.length();j++) {

char t = sb.charAt(j);

arrOfCount[(int)t-97]++;

}

for(int i=0;i<26;i++) {

int index;

if(arrOfCount[i]>=turn) {

while((index=sb.indexOf(String.valueOf((char)(i+97)))) != -1) {

sb.deleteCharAt(index);

}

}

}

if(sb.length()==0)

System.out.println("NO");

else {

char[] a = new char[sb.length()];

sb.getChars(0, sb.length(), a, 0);

LinkedList<Character> l = new LinkedList<Character>();

for(char ch : a)

l.add(ch);

Collections.sort(l);

l.forEach(e -> System.out.print(e));

}

}

}

**Alphabet Game 4**

import java.util.Collections;

import java.util.LinkedList;

import java.util.Scanner;

public class JavaAssignment {

public static void main(String[] args) {

Scanner kb = new Scanner(System.in);

StringBuffer sb = new StringBuffer(kb.next());

kb.close();

int[] arrOfCount = new int[26];

for(int j=0;j<sb.length();j++) {

char t = sb.charAt(j);

arrOfCount[(int)t-97]++;

}

int len = 0;

for(int i=0;i<sb.length();i++) {

if(arrOfCount[i]!=0)

len++;

}

LinkedList<Integer> list = new LinkedList<>();

for(int i=0;i<26;i++) {

int c = arrOfCount[i];

if(c!=0)

list.add(c);

}

Collections.sort(list);

while(len!=0) {

int c = list.remove();

for(int i=0;i<26;i++) {

int index;

if(arrOfCount[i]==c) {

while((index=sb.indexOf(String.valueOf((char)(i+97)))) != -1) {

sb.deleteCharAt(index);

arrOfCount[i]--;

}

len--;

if((i+1)<26 && arrOfCount[i+1]!=0) {

sb.deleteCharAt(sb.indexOf(String.valueOf((char)(i+98))));

arrOfCount[i+1]--;

len--;

}

if((i-1)>-1 && arrOfCount[i-1]!=0) {

System.out.println("char = "+(char)(i+96));

sb.deleteCharAt(sb.indexOf(String.valueOf((char)(i+96))));

arrOfCount[i]--;

len--;

}

}

if(len==0)

break;

}

}

if(sb.length()==0)

System.out.println("NO");

else {

char[] a = new char[sb.length()];

sb.getChars(0, sb.length(), a, 0);

LinkedList<Character> l = new LinkedList<Character>();

for(char ch : a)

l.add(ch);

Collections.sort(l);

l.forEach(e -> System.out.print(e));

}

}

}

**SMS Language**

public class JavaAssignment {

public static void main(String[] args) {

// TODO Auto-generated method stub

java.util.Scanner kb = new java.util.Scanner(System.in);

String s = kb.nextLine();

s = s.replaceAll("yes","s");

s = s.replaceAll("you", "u");

s = s.replaceAll("today", "2day");

s = s.replaceAll("why", "y");

System.out.println(s);

}

}

**TYPE 2**

**Comparable - Purchased Items**

class Item implements Comparable<Item>{

private String name;

private double price;

public Item(String name, double price) {

this.name = name;

this.price = price;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

@Override

public int compareTo(Item i) {

return price<i.price?1:price>i.price?-1:0;

}}

public class Main {

public static void main(String[] args) {

// TODO Auto-generated method stub

java.util.Scanner kb = new java.util.Scanner(System.in);

System.out.println("Please provide the number of items purchased");

int n = kb.nextInt();

java.util.ArrayList<Item> al = new java.util.ArrayList<>();

for(int i=1;i<=n;i++) {

System.out.println("Enter the name of the item : "+i);

String name = kb.next();

System.out.println("Enter the price of the item : "+i);

double price = kb.nextDouble();

al.add(new Item(name,price));

}

java.util.Collections.sort(al);

for(int i=0;i<n;i++)

System.out.println(i+1+" "+al.get(i).getName()+" "+al.get(i).getPrice());

}}

**Comparable - List Complaint By Date**

import java.time.LocalDate;

class Complaint implements Comparable<Complaint>{

private String customer;

private String complaintText;

private LocalDate dateOfComplaint;

private Integer severity;

public Complaint(String customer, String complaintText, LocalDate dateOfComplaint, Integer severity) {

super();

this.customer = customer;

this.complaintText = complaintText;

this.dateOfComplaint = dateOfComplaint;

this.severity = severity;

}

public String getCustomer() {

return customer;

}

public void setCustomer(String customer) {

this.customer = customer;

}

public String getComplaintText() {

return complaintText;

}

public void setComplaintText(String complaintText) {

this.complaintText = complaintText;

}

public LocalDate getDateOfComplaint() {

return dateOfComplaint;

}

public void setDateOfComplaint(LocalDate dateOfComplaint) {

this.dateOfComplaint = dateOfComplaint;

}

@Override

public int compareTo(Complaint i) {

return dateOfComplaint.isBefore(i.dateOfComplaint)?1:dateOfComplaint.isAfter(i.dateOfComplaint)?-1:0;

}

}

public class Main {

public static void main(String[] args) {

// TODO Auto-generated method stub

java.util.Scanner kb = new java.util.Scanner(System.in);

System.out.println("Please provide the number of complaints to be registered");

int n = kb.nextInt();

java.util.ArrayList<Complaint> al = new java.util.ArrayList<>();

for(int i=0;i<n;i++) {

System.out.println("Please provide your name");

String customer = kb.next();

System.out.println("Type in your complaint");

kb.nextLine();

String complaintText = kb.nextLine();

System.out.println("Please select the severity of your complaint\r\n" +

"1.Low\r\n" +

"2.Medium\r\n" +

"3.High");

Integer severity = kb.nextInt();

System.out.println("Please provide Date in (MM-dd-yyyy)");

java.time.LocalDate dateOfComplaint =

java.time.LocalDate.parse(kb.next(), java.time.format.DateTimeFormatter.ofPattern("M-d-y"));

al.add(new Complaint(customer,complaintText,dateOfComplaint,severity));

}

java.util.Collections.sort(al);

System.out.println("Complaint Details");

for(int i=0;i<n;i++)

System.out.println("Name "+al.get(i).getCustomer()

+"\nComplaint raised on "+al.get(i).getDateOfComplaint()

+"\n"+al.get(i).getComplaintText());

}

}

**Comparable - Display State**

**NOT WORKING**

import java.util.Collections;

import java.util.LinkedList;

import java.util.List;

import java.util.Map;

import java.util.Map.Entry;

import java.util.Scanner;

import java.util.TreeMap;

public class Main {

public static void main(String[] args) {

// TODO Auto-generated method stub

Map<Country, Country> mc=new TreeMap<>();

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

while(n-- > 0) {

String[] str = sc.next().split("\\|");

Country c = new Country(str[0]);

if(mc.containsKey(c)) {

c = mc.get(c);

c.addState(str[1]);

mc.put(c, c);

}

else {

c.addState(str[1]);

mc.put(c, c);

}

}

for(Entry<Country,Country> e : mc.entrySet()) {

Country c = e.getKey();

System.out.println(c+" "+c.getStateList());

}

}

}

class State implements Comparable<State> {

private String name;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public State(String name) {

super();

this.name = name;

}

@Override

public int compareTo(State o) {

// TODO Auto-generated method stub

return name.compareTo(o.name);

}

public String toString() {

return name;

}

}

class Country implements Comparable<Country> {

private String name;

List<State> li = new LinkedList<>();

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public List<State> getLi() {

return li;

}

public void setLi(List<State> li) {

this.li = li;

}

public Country(String name) {

super();

this.name = name;

}

public void addState(String stateName)

{

li.add(new State(stateName));

}

public List<State> getStateList()

{

Collections.sort(li);

return li;

}

@Override

public int hashCode() {

return name.hashCode();

}

@Override

public boolean equals(Object obj) {

Country c=(Country)obj;

return this.hashCode()==c.hashCode();

}

public String toString() {

return name;

}

@Override

public int compareTo(Country o) {

// TODO Auto-generated method stub

return name.compareTo(o.name);

}

}

**Comparable - Contact Information Based on Mobile Number**

import java.util.Collections;

import java.util.LinkedList;

import java.util.Scanner;

class Contact implements Comparable<Contact>{

private String name;

private String email;

private long mobile;

private String address;

public Contact(String name, String email, long mobile, String address) {

this.name = name;

this.email = email;

this.mobile = mobile;

this.address = address;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public long getMobile() {

return mobile;

}

public void setMobile(long mobile) {

this.mobile = mobile;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

@Override

public int compareTo(Contact c) {

return (int) (mobile - c.mobile);

}

}

public class Main {

public static void main(String[] args) {

Scanner kb = new Scanner(System.in);

System.out.print("Enter number of contacts : ");

int n = kb.nextInt();

LinkedList<Contact> list = new LinkedList<Contact>();

for(int i=1;i<=n;i++) {

System.out.println("Enter contact "+i+" detail");

System.out.print("Enter Name : ");

String name = kb.next();

System.out.print("Enter Email : ");

String email = kb.next();

System.out.print("Enter Mobile : ");

long mobile = kb.nextLong();

System.out.print("Enter Address : ");

String address = kb.next();

list.add(new Contact(name, email, mobile, address));

}

Collections.sort(list);

System.out.println("Contact list after sort by mobile number in descending order");

for(Contact c : list)

System.out.println(c.getName()+"-"+c.getMobile());

}

}

**Comparator - Complaint List Based on Severity and Date of Complaint**

import java.time.LocalDate;

import java.util.Comparator;

class Complaint{

private String customer;

private String complaintText;

private LocalDate dateOfComplaint;

private int severity;

public Complaint(String customer, String complaintText, LocalDate dateOfComplaint, int severity) {

this.customer = customer;

this.complaintText = complaintText;

this.dateOfComplaint = dateOfComplaint;

this.severity = severity;

}

public String getCustomer() {

return customer;

}

public String getComplaintText() {

return complaintText;

}

public LocalDate getDateOfComplaint() {

return dateOfComplaint;

}

public int getSeverity() {

return severity;

}

}

class ComplaintSeverityDateComparator implements Comparator<Complaint>{

@Override

public int compare(Complaint c1, Complaint c2) {

if(c1.getDateOfComplaint().isBefore(c2.getDateOfComplaint()))

return -1;

if(c1.getDateOfComplaint().isAfter(c2.getDateOfComplaint()))

return 1;

else

return c2.getSeverity()-c1.getSeverity();

}

}

public class Main {

public static void main(String[] args) {

// TODO Auto-generated method stub

java.util.Scanner kb = new java.util.Scanner(System.in);

System.out.println("Please provide the number of complaints to be registered");

int n = kb.nextInt();

java.util.ArrayList<Complaint> al = new java.util.ArrayList<>();

for(int i=0;i<n;i++) {

System.out.println("Please provide your name");

String customer = kb.next();

System.out.println("Type in your complaint");

kb.nextLine();

String complaintText = kb.nextLine();

System.out.println("Please select the severity of your complaint\r\n" +

"1.Low\r\n" +

"2.Medium\r\n" +

"3.High");

Integer severity = kb.nextInt();

System.out.println("Please provide Date in (MM-dd-yyyy)");

java.time.LocalDate dateOfComplaint =

java.time.LocalDate.parse(kb.next(), java.time.format.DateTimeFormatter.ofPattern("M-d-y"));

al.add(new Complaint(customer,complaintText,dateOfComplaint,severity));

}

java.util.Collections.sort(al, new ComplaintSeverityDateComparator());

System.out.println("Complaint Details");

for(int i=0;i<n;i++)

System.out.println("Name "+al.get(i).getCustomer()

+"\nComplaint raised on "+al.get(i).getDateOfComplaint()

+"\n"+al.get(i).getComplaintText());

}

}

**TYPE 3**

**TreeMap - Letter Frequency**

import java.util.Map.Entry;

import java.util.Scanner;

import java.util.TreeMap;

class LetterSequence{

private String sentence;

public LetterSequence(String sentence) {

this.sentence = sentence;

}

public String getSentence() {

return sentence;

}

public void setSentence(String sentence) {

this.sentence = sentence;

}

public TreeMap<Character,Integer>computeFrequency() {

TreeMap<Character,Integer> tm = new TreeMap<Character, Integer>();

int len = sentence.length();

for(int i=0;i<len;i++) {

char c = sentence.charAt(i);

if(c==' ')

continue;

if(tm.containsKey(c)) {

int count = tm.get(c);

tm.put(c, ++count);

}

else {

tm.put(c,1);

}

}

return tm;

}

public void displayLetterFrequency(TreeMap<Character,Integer>frequencyMap) {

for(Entry<Character,Integer> e : frequencyMap.entrySet()) {

System.out.print(e.getKey()+" : ");

int i = e.getValue();

while(i-- > 0)

System.out.print("\*");

System.out.println();

}

}

}

public class Main {

public static void main(String[] args) {

Scanner kb = new Scanner(System.in);

LetterSequence ls = new LetterSequence(kb.nextLine());

ls.displayLetterFrequency(ls.computeFrequency());

}

}

**HashMap - Complaint Register**

import java.util.HashMap;

import java.util.Scanner;

class User{

private String name;

private int complaintCount;

public User(String name, int complaintCount) {

this.name = name;

this.complaintCount = complaintCount;

}

public String toString() {

return "User name : "+name+"\nComplaint Count : "+complaintCount;

}

}

public class Main {

public static void main(String[] args) {

Scanner kb = new Scanner(System.in);

HashMap<String,User> hm = new HashMap<>();

String ch = null;

do {

System.out.println("Enter the user name");

String name = kb.next();

System.out.println("Enter complaints - separated by \"|\" symbol");

kb.nextLine();

String s = kb.nextLine();

hm.put(name, new User(name,s.split("\\|").length));

System.out.println("Do you want to add another user (yes/no)");

ch = kb.next();

}while(!ch.equals("no"));

do {

System.out.println("Enter the user name to search");

String name = kb.next();

User u = hm.get(name);

if(u == null)

System.out.println("No user found with the name "+name);

else

System.out.println(u);

System.out.println("Do you want to search another user (yes/no)");

ch = kb.next();

}while(!ch.equals("no"));

}

}

**HashMap - Complaint Register View**

import java.util.HashMap;

import java.util.LinkedList;

import java.util.List;

import java.util.Scanner;

class User{

private String name;

public User(String name) {

this.name = name;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String toString() {

return name;

}

@Override

public int hashCode() {

return name.hashCode();

}

@Override

public boolean equals(Object o) {

User u = (User) o;

return name.equals(u.name);

}

}

class Complaint{

private String description;

private User user;

public Complaint(String description, User user) {

this.description = description;

this.user = user;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public User getUser() {

return user;

}

public void setUser(User user) {

this.user = user;

}

public String toString() {

return description;

}

}

public class Main {

public static void main(String[] args) {

Scanner kb = new Scanner(System.in);

HashMap<User,List<Complaint>> hm = new HashMap<>();

LinkedList<Complaint> list = new LinkedList<Complaint>();

String ch = null;

do {

System.out.println("Enter the user name");

String name = kb.next();

System.out.println("Enter complaints - separated by \"|\" symbol");

kb.nextLine();

String[] s = kb.nextLine().split("\\|");

User u = new User(name);

list.clear();

for(String t : s)

list.add(new Complaint(t,u));

hm.put(u,list);

System.out.println("Do you want to add another user (yes/no)");

ch = kb.next();

}while(!ch.equals("no"));

do {

System.out.println("Enter the user name to search");

String name = kb.next();

List<Complaint> l = hm.get(new User(name));

if(l == null)

System.out.println("No user found with the name "+name);

else {

System.out.println("Complaints of user "+name);

for(int i=0;i<l.size();i++) {

System.out.println(i+1+") "+l.get(i));

}

}

System.out.println("Do you want to search another user (yes/no)");

ch = kb.next();

}while(!ch.equals("no"));

}

}

**HashMap - Currency Conversion**

public class Main {

public static void main(String[] args) {

java.util.Scanner kb = new java.util.Scanner(System.in);

System.out.println("Enter INR");

double i = kb.nextDouble();

int ch = 0;

java.util.HashMap<String, Double> hm = new java.util.HashMap<String, Double>();

hm.put("US Dollar",0.0149);

hm.put("Euro",0.0135);

hm.put("Yen",1.5211);

outer:while(ch!=4) {

System.out.println("1.US Dollar"

+ "\n2.Euro"

+ "\n3.Yen"

+ "\n4.Exit");

System.out.println("Enter Choice");

ch = kb.nextInt();

switch(ch) {

case 1:

System.out.printf("Dollar value : %.2f\n",i\*hm.get("US Dollar"));

break;

case 2:

System.out.printf("Euro value : %.2f\n",i\*hm.get("Euro"));

break;

case 3:

System.out.printf("Yen value : %.2f\n",i\*hm.get("Yen"));

break;

case 4:

break outer;

default :

System.out.println("Invalid option");

}

}

kb.close();

}

}

**HashMap - Numbers to Bin**

class Histogram{

private HashMap<Integer, Integer> bins;

public Histogram(HashMap<Integer, Integer> bins) {

this.bins = bins;

bins.put(10,0);

bins.put(20,0);

bins.put(30,0);

bins.put(40,0);

}

public void addNumber(Integer i) {

if(i>=1 && i<=10) {

int count = bins.get(10);

bins.put(10, ++count);

}

else if(i>=11 && i<=20) {

int count = bins.get(20);

bins.put(20, ++count);

}

else if(i>=21 && i<=30) {

int count = bins.get(30);

bins.put(30, ++count);

}

else if(i>=31 && i<=40) {

int count = bins.get(40);

bins.put(40, ++count);

}

}

public void displayHistogram() {

System.out.println("Histogram");

for(int i=10;i<41;i=i+10) {

System.out.print(i+" : ");

int c = bins.get(i);

while(c-- > 0)

System.out.print("\*");

System.out.println();

}

}

}

public class Main {

public static void main(String[] args) {

java.util.Scanner kb = new java.util.Scanner(System.in);

int n = kb.nextInt();

Histogram h = new Histogram(new HashMap<Integer, Integer>());

while(n-- > 0) {

h.addNumber(kb.nextInt());

}

kb.close();

h.displayHistogram();

}

}

**TYPE 4**

**Wicket details**

class Wicket{

private long over;

private long ball;

private String wicketType;

private String playerName;

private String bowlerName;

public Wicket(long over, long ball, String wicketType, String playerName, String bowlerName) {

this.over = over;

this.ball = ball;

this.wicketType = wicketType;

this.playerName = playerName;

this.bowlerName = bowlerName;

}

public String getWicketType() {

return wicketType;

}

public String toString() {

return String.format("\nOver : %d"

+ "\nBall : %d"

+ "\nWicket Type : %s"

+ "\nPlayer Name : %s"

+ "\nBowler Name : %s", over, ball, wicketType, playerName, bowlerName);

}

}

class WicketBO{

public static void displayAllWicketDetails(Wicket[] wicketList) {

System.out.println("Wicket Details");

int i = 1;

for(Wicket w : wicketList)

System.out.println("Wicket "+(i++)+w);

}

public static void displaySpecificWicketDetails(Wicket[] wicketList, String wicketType) {

int i = 1;

for(Wicket w : wicketList) {

if(w.getWicketType().equals(wicketType))

System.out.println("Wicket "+(i++)+w);

}

}

}

public class Main {

public static void main(String[] args) {

java.util.Scanner kb = new java.util.Scanner(System.in);

System.out.println("Enter the number of wickets");

int n = kb.nextInt();

Wicket[] w = new Wicket[n];

kb.nextLine();

for(int i=0;i<n;i++) {

System.out.println("Enter the details of wicket "+(i+1));

String[] s = kb.nextLine().split(",");

w[i] = new Wicket(Long.parseLong(s[0]),Long.parseLong(s[1]),s[2],s[3],s[4]);

}

WicketBO.displayAllWicketDetails(w);

System.out.println("Enter the wicket type to be searched");

WicketBO.displaySpecificWicketDetails(w, kb.next());

kb.close();

}

}

**Player Details**

class Player{

private String name;

private String country;

private String skill;

public Player(String name, String country, String skill) {

this.name = name;

this.country = country;

this.skill = skill;

}

public String getCountry() {

return country;

}

public String toString() {

return String.format("%-15s %-15s %s", name, country, skill);

}

}

class PlayerBO{

public static void displayAllPlayerDetails(Player[] playerList) {

System.out.println("Player Details");

for(Player p : playerList)

System.out.println(p);

}

public static void displaySpecificPlayerDetails(Player[] playerList, String countryName) {

for(Player p : playerList) {

if(p.getCountry().equals(countryName))

System.out.println(p);

}

}

}

public class Main {

public static void main(String[] args) {

java.util.Scanner kb = new java.util.Scanner(System.in);

System.out.println("Enter the number of players");

int n = kb.nextInt();

Player[] p = new Player[n];

kb.nextLine();

for(int i=0;i<n;i++) {

System.out.println("Enter the player name");

String name = kb.nextLine();

System.out.println("Enter the country name");

String country = kb.nextLine();

System.out.println("Enter the skill");

String skill = kb.nextLine();

p[i] = new Player(name, country, skill);

}

PlayerBO.displayAllPlayerDetails(p);

System.out.println("Enter the country name for which players details to be known");

PlayerBO.displaySpecificPlayerDetails(p, kb.nextLine());

kb.close();

}}

**Match Details**

class Match{

private String date;

private String teamOne;

private String teamTwo;

private String venue;

public Match(String date, String teamOne, String teamTwo, String venue) {

this.date = date;

this.teamOne = teamOne;

this.teamTwo = teamTwo;

this.venue = venue;

}

public String getDate() {

return date;

}

public String toString() {

return String.format("%-35s %-30s %-15s %s", teamOne, teamTwo, date, venue);

}

}

class MatchBO{

public static void displayAllMatchDetails(Match[] matchList) {

System.out.println("Match Details");

System.out.printf("%-35s %-30s %-15s %s\n","Team 1","Team 2","Date","Venue");

for(Match p : matchList)

System.out.println(p);

}

public static void displaySpecificMatchDetails(Match[] matchList, String date) {

System.out.println("Match Details");

System.out.printf("%-35s %-30s %-15s %s\n","Team 1","Team 2","Date","Venue");

for(Match p : matchList) {

if(p.getDate().equals(date))

System.out.println(p);

}

}

}

public class Main {

public static void main(String[] args) {

java.util.Scanner kb = new java.util.Scanner(System.in);

System.out.println("Enter the number of matches");

int n = kb.nextInt();

Match[] m = new Match[n];

kb.nextLine();

for(int i=0;i<n;i++) {

System.out.println("Enter the match "+(i+1)+"details");

System.out.println("Enter the match date");

String date = kb.nextLine();

System.out.println("Enter the team one");

String teamOne = kb.nextLine();

System.out.println("Enter the team two");

String teamTwo = kb.nextLine();

System.out.println("Enter the Venue");

String venue = kb.nextLine();

m[i] = new Match(date, teamOne, teamTwo, venue);

}

MatchBO.displayAllMatchDetails(m);

System.out.println("Enter the date to be searched");

MatchBO.displaySpecificMatchDetails(m, kb.nextLine());

kb.close();

}

}

**Outcome Details**

class Outcome{

private String date;

private String status;

private String winnerTeam;

private String playerOfMatch;

public Outcome(String date, String status, String winnerTeam, String playerOfMatch) {

this.date = date;

this.status = status;

this.winnerTeam = winnerTeam;

this.playerOfMatch = playerOfMatch;

}

public String getDate() {

return date;

}

public String toString() {

return String.format("%-20s %-20s %-120s %s", status, winnerTeam, playerOfMatch, date);

}

}

class OutcomeBO{

public static void displayAllOutcomeDetails(Outcome[] outcomeList) {

System.out.println("Outcome Details");

System.out.printf("%-20s %-20s %-20s %s\n","Status","Winning Team","Player Of The Match","Date");

for(Outcome p : outcomeList)

System.out.println(p);

}

public static void displaySpecificOutcomeDetails(Outcome[] outcomeList, String date) {

System.out.println("Outcome Details");

System.out.printf("%-20s %-20s %-20s %s\n","Status","Winning Team","Player Of The Match","Date");

for(Outcome p : outcomeList) {

if(p.getDate().equals(date))

System.out.println(p);

}

}

}

public class Main {

public static void main(String[] args) {

java.util.Scanner kb = new java.util.Scanner(System.in);

System.out.println("Enter the number of matches");

int n = kb.nextInt();

Outcome[] o = new Outcome[n];

kb.nextLine();

for(int i=0;i<n;i++) {

System.out.println("Enter the match "+(i+1)+" details");

System.out.println("Enter the date");

String date = kb.nextLine();

System.out.println("Enter the status");

String status = kb.nextLine();

System.out.println("Enter the winner team");

String winnerTeam = kb.nextLine();

System.out.println("Enter the player of match");

String playerOfMatch = kb.nextLine();

o[i] = new Outcome(date, status, winnerTeam, playerOfMatch);

}

OutcomeBO.displayAllOutcomeDetails(o);

System.out.println("Enter the date to be searched");

OutcomeBO.displaySpecificOutcomeDetails(o, kb.nextLine());

kb.close();

}

}

**Delivery Details**

class Delivery{

private long over;

private long ball;

private long runs;

private String nonStriker;

private String batsman;

private String bowler;

public Delivery(long over, long ball, long runs, String nonStriker, String batsman, String bowler) {

this.over = over;

this.ball = ball;

this.runs = runs;

this.nonStriker = nonStriker;

this.batsman = batsman;

this.bowler = bowler;

}

public long getOver() {

return over;

}

public long getBall() {

return ball;

}

public long getRuns() {

return runs;

}

public String getBatsman() {

return batsman;

}

public String getBowler() {

return bowler;

}

public String toString() {

return String.format("%-20s %-20s %-20s %-20s %-20s %s", over, ball, runs, batsman, bowler, nonStriker);

}

}

class DeliveryBO{

public static void displayAllOutcomeDetails(Delivery[] deliveryList) {

System.out.println("Delivery Details");

System.out.printf("%-20s %-20s %-20s %-20s %-20s %s"

, "Over", "Ball", "Runs", "Batsman", "Bowler", "NonStriker\n");

for(Delivery d : deliveryList)

System.out.println(d);

}

public static void displayBatsmanBowlerDetails(Delivery[] deliveryList, long ball, long over) {

for(Delivery d : deliveryList) {

if(d.getBall() == ball) {

System.out.println("Batsman : "+d.getBatsman()+"\nBowler : "+d.getBowler());

break;

}

}

}

public static void displayMaximumRunDetails(Delivery[] deliveryList) {

long max = 0;

for(Delivery d : deliveryList) {

long t = d.getRuns();

if(t>max)

max = t;

}

for(Delivery d : deliveryList) {

long runs = d.getRuns();

if(runs==max) {

System.out.println("Maximum Runs : "+runs

+"\nOver : "+d.getOver()

+"\nBall : "+d.getBall());

}

}

}

}

public class Main {

public static void main(String[] args) {

java.util.Scanner kb = new java.util.Scanner(System.in);

System.out.println("Enter the number of matches");

int n = kb.nextInt();

Delivery[] d = new Delivery[n];

for(int i=0;i<n;i++) {

System.out.println("Enter the over");

long over = kb.nextLong();

System.out.println("Enter the ball");

long ball = kb.nextLong();

System.out.println("Enter the runs");

long runs = kb.nextLong();

kb.nextLine();

System.out.println("Enter the batsman name");

String batsmanName = kb.nextLine();

System.out.println("Enter the bowler name");

String bowlerName = kb.nextLine();

System.out.println("Enter the non-striker name");

String nonStriker = kb.nextLine();

d[i] = new Delivery(over, ball, runs, nonStriker, batsmanName, bowlerName);

}

System.out.println("Enter the choice"

+ "\n1.View Details"

+ "\n2.Batsman and Bowler"

+ "\n3.Maximum runs");

switch(kb.nextInt()) {

case 1:

DeliveryBO.displayAllOutcomeDetails(d);

break;

case 2:

System.out.println("Enter the over for which batsman and bowler to be known");

int over = kb.nextInt();

System.out.println("Enter the ball for which batsman and bowler to be known");

int ball = kb.nextInt();

DeliveryBO.displayBatsmanBowlerDetails(d, ball, over);

break;

case 3:

DeliveryBO.displayMaximumRunDetails(d);

}

kb.close();

}

}

**TYPE 5**

**Trainee Class**

class Trainee{

private int id;

private String name;

private String batchName;

private int age;

public Trainee(int id, String batchName, String name, int age) {

super();

this.id = id;

this.batchName = batchName;

this.name = name;

this.age = age;

}

public String toString() {

return id+" "+name+" "+batchName+" "+age;

}

}

class IdRangeException extends Exception{

private String msg;

public IdRangeException() {}

public IdRangeException(String msg) {

super(msg);

this.msg = msg;

}

public String toString() {

return "IdRangeException: "+msg;

}

}

class AgeRangeException extends Exception{

private String msg;

public AgeRangeException() {}

public AgeRangeException(String msg) {

super(msg);

this.msg = msg;

}

public String toString() {

return "AgeRangeException: "+msg;

}

}

public class Main {

public static void main(String[] args) {

java.util.Scanner kb = new java.util.Scanner(System.in);

System.out.println("Enter the id of the trainee");

int id = 0;

boolean flag = false;

try {

id = Integer.parseInt(kb.next());

if(id<10000 || id>99999)

throw new IdRangeException("Id not within range");

}catch(IdRangeException i) {

do {

System.out.println(i);

System.out.println("Please enter the id again");

id = Integer.parseInt(kb.next());

}while(id<10000 || id>99999);

}catch(NumberFormatException n) {

do {

if(flag)

System.out.println("IdRangeException: Id not within range");

else

System.out.println(n);

System.out.println("Please enter the id again");

id = Integer.parseInt(kb.next());

flag = true;

}while(id<10000 || id>99999);

}

System.out.println("Enter the name of the trainee");

String name = kb.next();

System.out.println("Enter the batch name of the trainee");

String batchName = kb.next();

System.out.println("Enter the age of the trainee");

int age = 0;

flag = false;

try {

age = Integer.parseInt(kb.next());

if(age<20 || age>25)

throw new AgeRangeException("Age not within range");

}catch(AgeRangeException a) {

do {

System.out.println(a);

System.out.println("Please enter the age again");

age = Integer.parseInt(kb.next());

}while(age<20 || age>25);

}catch(NumberFormatException n) {

do {

if(flag)

System.out.println("AgeRangeException: Age not within range");

else

System.out.println(n);

System.out.println("Please enter the age again");

age = Integer.parseInt(kb.next());

flag = true;

}while(age<20 || age>25);

}

System.out.println("Trainee details are\n"+new Trainee(id, name, batchName, age));

kb.close();

}

}

**Batch Class**

class Batch{

private String name;

private String trainerName;

private int batchSize;

private int averageMarks;

public Batch(String name, String trainerName, int batchSize, int averageMarks) {

super();

this.name = name;

this.trainerName = trainerName;

this.batchSize = batchSize;

this.averageMarks = averageMarks;

}

public String toString() {

return name+" "+trainerName+" "+batchSize+" "+averageMarks;

}

}

class BatchSizeException extends Exception{

private String msg;

public BatchSizeException () {}

public BatchSizeException (String msg) {

super(msg);

this.msg = msg;

}

public String toString() {

return "BatchSizeException: "+msg;

}

}

class AverageMarksException extends Exception{

private String msg;

public AverageMarksException() {}

public AverageMarksException(String msg) {

super(msg);

this.msg = msg;

}

public String toString() {

return "AverageMarksException: "+msg;

}

}

public class Main {

public static void main(String[] args) {

java.util.Scanner kb = new java.util.Scanner(System.in);

System.out.println("Enter the name of the batch");

String batchName = kb.next();

System.out.println("Enter the trainer name of the batch");

String name = kb.next();

System.out.println("Enter the size of the batch");

int batchSize = 0;

boolean flag = false;

try {

batchSize = Integer.parseInt(kb.next());

if(batchSize<10 || batchSize>30)

throw new BatchSizeException("Batch size not within range");

}catch(BatchSizeException b) {

do {

System.out.println(b);

System.out.println("Please enter the batch size again");

batchSize = Integer.parseInt(kb.next());

}while(batchSize<10 || batchSize>30);

}catch(NumberFormatException n) {

do {

if(flag)

System.out.println("BatchSizeException: Batch size not within range");

else

System.out.println(n);

System.out.println("Please enter the batch size again");

batchSize = Integer.parseInt(kb.next());

flag = true;

}while(batchSize<10 || batchSize>30);

}

System.out.println("Enter the average marks of the batch");

int averageMarks = 0;

flag = false;

try {

averageMarks = Integer.parseInt(kb.next());

if(averageMarks<0 || averageMarks>100)

throw new AverageMarksException("Average marks not within range");

}catch(AverageMarksException a) {

do {

System.out.println(a);

System.out.println("Please enter the average marks again");

averageMarks = Integer.parseInt(kb.next());

}while(averageMarks<0 || averageMarks>100);

}catch(NumberFormatException n) {

do {

if(flag)

System.out.println("AverageMarksException: Average marks not within range");

else

System.out.println(n);

System.out.println(n);

System.out.println("Please enter the average marks again");

averageMarks = Integer.parseInt(kb.next());

flag = true;

}while(averageMarks<0 || averageMarks>100);

}

System.out.println("The batch details are\n"+new Batch(name, batchName, batchSize, averageMarks));

kb.close();

}

}

**Trainer Class**

class Trainer{

private int id;

private String name;

private String stream;

private int noOfBatches;

public Trainer(int id, String name, String stream, int noOfBatches) {

this.id = id;

this.name = name;

this.stream = stream;

this.noOfBatches = noOfBatches;

}

public String toString() {

return id+" "+name+" "+stream+" "+noOfBatches;

}

}

class IdRangeException extends Exception{

private String msg;

public IdRangeException () {}

public IdRangeException (String msg) {

super(msg);

this.msg = msg;

}

public String toString() {

return "IdRangeException: "+msg;

}

}

class InvalidStreamException extends Exception{

private String msg;

public InvalidStreamException() {}

public InvalidStreamException(String msg) {

super(msg);

this.msg = msg;

}

public String toString() {

return "InvalidStreamException: "+msg;

}

}

public class Main {

public static void main(String[] args) {

java.util.Scanner kb = new java.util.Scanner(System.in);

System.out.println("Enter the id of the trainer");

int id = 0;

boolean flag = false;

try {

id = Integer.parseInt(kb.next());

if(id<10000 || id>99999)

throw new IdRangeException("Id not within range");

}catch(IdRangeException i) {

do {

System.out.println(i);

System.out.println("Please enter the id again");

id = Integer.parseInt(kb.next());

}while(id<10000 || id>99999);

}catch(NumberFormatException n) {

do {

if(flag)

System.out.println("IdRangeException: Id not within range");

else

System.out.println(n);

System.out.println("Please enter the id again");

id = Integer.parseInt(kb.next());

flag = true;

}while(id<10000 || id>99999);

}

System.out.println("Enter the name of the trainer");

String name = kb.next();

System.out.println("Enter the stream of the trainer");

String stream = null;

try {

stream = kb.next();

if(!stream.matches("Java|CSharp|Mainframes"))

throw new InvalidStreamException("An invalid stream");

}catch(InvalidStreamException a) {

do {

System.out.println(a);

System.out.println("Please enter the stream again");

stream = kb.next();

}while(!stream.matches("Java|CSharp|Mainframes"));

}

System.out.println("Enter the number of batches allotted");

int noOfBatches = kb.nextInt();

System.out.println("The trainer details are\n" + new Trainer(id, name, stream, noOfBatches));

kb.close();

}

}

**BatchOwner Class**

class Trainer{

private int id;

private String name;

private String location;

private int noOfBatches;

public Trainer(int id, String name, String location, int noOfBatches) {

this.id = id;

this.name = name;

this.location = location;

this.noOfBatches = noOfBatches;

}

public String toString() {

return id+" "+name+" "+location+" "+noOfBatches;

}

}

class IdRangeException extends Exception{

private String msg;

public IdRangeException () {}

public IdRangeException (String msg) {

super(msg);

this.msg = msg;

}

public String toString() {

return "IdRangeException: "+msg;

}

}

class AllottedBatchesException extends Exception{

private String msg;

public AllottedBatchesException() {}

public AllottedBatchesException(String msg) {

super(msg);

this.msg = msg;

}

public String toString() {

return "AllottedBatchesException: "+msg;

}

}

public class Main {

public static void main(String[] args) {

java.util.Scanner kb = new java.util.Scanner(System.in);

System.out.println("Enter the id of the batch owner");

int id = 0;

boolean flag = false;

try {

id = kb.nextInt();

if(id<10000 || id>99999)

throw new IdRangeException("Id not within range");

}catch(IdRangeException i) {

do {

System.out.println(i);

System.out.println("Please enter the id again");

id = kb.nextInt();

}while(id<10000 || id>99999);

}catch(NumberFormatException n) {

do {

if(flag)

System.out.println("IdRangeException: Id not within range");

else

System.out.println(n);

System.out.println("Please enter the id again");

id = kb.nextInt();

flag = true;

}while(id<10000 || id>99999);

}

System.out.println("Enter the name of the batch owner");

String name = kb.next();

System.out.println("Enter the location of the batch owner");

String location = kb.next();

System.out.println("Enter the number of batches allotted");

int noOfBatches = 0;

flag = false;

try {

noOfBatches = Integer.parseInt(kb.next());

if(noOfBatches<5 || noOfBatches>10)

throw new AllottedBatchesException("Number of batches allotted not within range");

}catch(AllottedBatchesException a) {

do {

System.out.println(a);

System.out.println("Please enter the number of batches allotted again");

noOfBatches = Integer.parseInt(kb.next());

}while(noOfBatches<5 || noOfBatches>10);

}catch(NumberFormatException n) {

do {

if(flag)

System.out.println("AllottedBatchesException: Number of batches allotted not within range");

else

System.out.println(n);

System.out.println("Please enter the number of batches allotted again");

noOfBatches = Integer.parseInt(kb.next());

flag = true;

}while(noOfBatches<5 || noOfBatches>10);

}

System.out.println("The batch owner details are\n" + new Trainer(id, name, location, noOfBatches));

kb.close();

}

}

**Course Class**

class Course{

private int id;

private String name;

private String type;

private int duration;

public Course(int id, String name, String type, int duration) {

this.id = id;

this.name = name;

this.type = type;

this.duration = duration;

}

public String toString() {

return id+" "+name+" "+type+" "+duration;

}

}

class IdRangeException extends Exception{

private String msg;

public IdRangeException () {}

public IdRangeException (String msg) {

super(msg);

this.msg = msg;

}

public String toString() {

return "IdRangeException: "+msg;

}

}

class InvalidTypeException extends Exception{

private String msg;

public InvalidTypeException() {}

public InvalidTypeException(String msg) {

super(msg);

this.msg = msg;

}

public String toString() {

return "InvalidTypeException: "+msg;

}

}

public class Main {

public static void main(String[] args) {

java.util.Scanner kb = new java.util.Scanner(System.in);

System.out.println("Enter the id of the course");

int id = 0;

boolean flag = false;

try {

id =Integer.parseInt(kb.next());

if(id<10000 || id>99999)

throw new IdRangeException("Id not within range");

}catch(IdRangeException i) {

do {

System.out.println(i);

System.out.println("Please enter the id again");

id = Integer.parseInt(kb.next());

}while(id<10000 || id>99999);

}catch(NumberFormatException n) {

do {

if(flag)

System.out.println("IdRangeException: Id not within range");

else

System.out.println(n);

System.out.println("Please enter the id again");

id = Integer.parseInt(kb.next());

flag = true;

}while(id<10000 || id>99999);

}

System.out.println("Enter the name of the course");

String name = kb.next();

System.out.println("Enter the type of the course");

String type = null;

try {

type = kb.next();

if(!type.matches("HandsOn|Theory|SelfLearning"))

throw new InvalidTypeException("Invalid Type");

}catch(InvalidTypeException a) {

do {

System.out.println(a);

System.out.println("Please enter the type again");

type = kb.next();

}while(!type.matches("HandsOn|Theory|SelfLearning"));

}

System.out.println("Enter the duration of the course");

int duration = kb.nextInt();

System.out.println("The batch owner details are\n" + new Course(id, name, type, duration));

kb.close();

}

}

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

class UserMainCode{

public static String getString(String s) {

StringBuffer sb = new StringBuffer(s);

if(sb.charAt(0)!='j' && sb.charAt(1)=='b') {

sb.deleteCharAt(0);

}

else if(sb.charAt(0)=='j' && sb.charAt(1)!='b') {

sb.deleteCharAt(1);

}

else if(sb.charAt(0)!='j' && sb.charAt(1)!='b')

sb.deleteCharAt(0).deleteCharAt(0);

return String.valueOf(sb);

}

}

public class Main {

public static void main(String[] args) {

java.util.Scanner kb = new java.util.Scanner(System.in);

System.out.println(UserMainCode.getString(kb.next()));

kb.close();

}

}

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

class UserMainCode{

public static int validateColorCode(String s) {

if(s.matches("#[[A-F]|[0-9]]{6}"))

return 1;

return -1;

}

}

public class Main {

public static void main(String[] args) {

java.util.Scanner kb = new java.util.Scanner(System.in);

System.out.println(UserMainCode.validateColorCode(kb.next())==1?"Valid":"Invalid");

kb.close();

}

}

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

class UserMainCode{

public static int getDigitSum(int i) {

if(i<0)

return -1;

if(i>=0 && i<=9)

return i;

int len = String.valueOf(i).length();

int sum = 0;

while(len-- > 0) {

sum = sum + i%10;

i = i/10;

}

len = String.valueOf(sum).length();

if(len!=1)

sum = getDigitSum(sum);

return sum;

}

}

public class Main {

public static void main(String[] args) {

java.util.Scanner kb = new java.util.Scanner(System.in);

System.out.println(UserMainCode.getDigitSum(kb.nextInt()));

kb.close();

}

}

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

class UserMainCode{

private static boolean isLeapYear(int year) {

if((year%4==0 && year%100!=0) || (year%400==0))

return true;

return false;

}

public static int getNumberOfDays(int year, int month) {

if(month==2) {

if(isLeapYear(year))

return 29;

else

return 28;

}

if(month==1 || month==3 || month==5 || month==7 || month==8 || month==10 || month==12)

return 31;

return 30;

}

}

public class Main {

public static void main(String[] args) {

java.util.Scanner kb = new java.util.Scanner(System.in);

System.out.println(UserMainCode.getNumberOfDays(kb.nextInt(), kb.nextInt()));

}

}

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

class UserMainCode{

public static int addOddNumbers(int num) {

int n = 0;

if(num%2==0) {

n = num/2;

num--;

}

else

n = (num-1)/2 + 1;

return n\*(num+1)/2;

}

}

public class Main {

public static void main(String[] args) {

java.util.Scanner kb = new java.util.Scanner(System.in);

System.out.println(UserMainCode.addOddNumbers(kb.nextInt()));

}

}

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

class UserMainCode{

public static String[] convertToStringArray (java.util.ArrayList<String> al) {

String[] s = new String[al.size()];

int i = 0;

for(String x : al)

s[i++] = x;

return s;

}

}

public class Main {

public static void main(String[] args) {

java.util.Scanner kb = new java.util.Scanner(System.in);

java.util.ArrayList<String> al = new java.util.ArrayList<>();

String[] s = UserMainCode.convertToStringArray(al);

for(String x : s)

System.out.println(x);

}

}

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

class UserMainCode{

public static int commonChars(String s1, String s2) {

StringBuffer sb1 = new StringBuffer(s1);

StringBuffer sb2 = new StringBuffer(s2);

while(sb1.indexOf(" ") != -1) {

sb1.deleteCharAt(sb1.indexOf(" "));

}

while(sb2.indexOf(" ") != -1) {

sb2.deleteCharAt(sb2.indexOf(" "));

}

int[] arrCount1 = new int[52];

int[] arrCount2 = new int[52];

int len1 = sb1.length();

int len2 = sb2.length();

for(int i=0;i<len1;i++) {

char c = sb1.charAt(i);

int index = (int)c-97;

if(index<0)

arrCount1[(int)c-65]++;

arrCount1[(int)c-71]++;

}

for(int i=0;i<len2;i++) {

char c = sb2.charAt(i);

int index = (int)c-97;

if(index<0)

arrCount2[(int)c-65]++;

arrCount2[(int)c-71]++;

}

int count = 0;

for(int i=0;i<len1;i++) {

char c1 = sb1.charAt(i);

int index = (int)c1 - 97;

if(index<0)

index = (int)c1 - 65;

index = (int)c1 - 71;

for(int j=0;j<len2;j++) {

char c2 = sb2.charAt(j);

if(c1==c2 && arrCount1[index]==1 && arrCount2[index]==1) {

count++;

}

}

}

return count;

}

}

public class Main {

public static void main(String[] args) {

java.util.Scanner kb = new java.util.Scanner(System.in);

System.out.println(UserMainCode.commonChars(kb.nextLine(),kb.nextLine()));

}

}

**EmployeesList**

import java.time.LocalDate;

import java.time.format.DateTimeFormatter;

import java.util.Collections;

import java.util.LinkedList;

import java.util.Scanner;

class Employee implements Comparable<Employee>{

private static int counter = 0;

private int id;

private String name;

private String department;

private LocalDate dateOfJoining;

private int age;

private int salary;

public Employee(String name, String department, LocalDate dateOfJoining, int age, int salary) {

this.id = ++counter;

this.name = name;

this.department = department;

this.dateOfJoining = dateOfJoining;

this.age = age;

this.salary = salary;

}

public LocalDate getDateOfJoining() {

return dateOfJoining;

}

public int getAge() {

return age;

}

@Override

public String toString() {

return String.format("%-15s %-30s %-30s %-10s %-10s %-10s %-10s\n", id,name,department,dateOfJoining,age,salary);

}

@Override

public int compareTo(Employee e) {

return salary-e.salary;

}

}

class AgeComparator implements java.util.Comparator<Employee>{

@Override

public int compare(Employee e1, Employee e2) {

if(e1.getAge()<e2.getAge())

return -1;

if(e1.getAge()>e2.getAge())

return 1;

else {

if(e1.getDateOfJoining().isBefore(e2.getDateOfJoining()))

return -1;

if(e1.getDateOfJoining().isAfter(e2.getDateOfJoining()))

return 1;

}

return 0;

}

}

class EmployeeBO{

public static void printEmployees(LinkedList<Employee> emp) {

System.out.println(String.format("%-15s %-30s %-30s %-10s %-10s %-10s %-10s\n"

,"Employee ID","Name","Department","Date Of Joining","Age","Salary"));

for(Employee e : emp)

System.out.println(e);

}

}

public class Main {

public static void main(String[] args) {

Scanner kb = new Scanner(System.in);

System.out.println("Input the number of employees :");

int n = kb.nextInt();

LinkedList<Employee> list = new LinkedList<>();

for(int i=0;i<n;i++) {

System.out.println("Enter the details for employee "+(i+1));

String name = kb.next();

kb.nextLine();

String department = kb.nextLine();

LocalDate date = LocalDate.parse(kb.next(), DateTimeFormatter.ofPattern("M/d/y"));

int age = kb.nextInt();

int salary = kb.nextInt();

list.add(new Employee(name, department, date, age, salary));

}

System.out.println("1.Sort employees by salary\r\n" +

"2.Sort employees by age and by date of joining\r\n" +

"Enter your choice\r\n");

int ch = kb.nextInt();

kb.close();

if(ch==1)

Collections.sort(list);

else

Collections.sort(list, new AgeComparator());

EmployeeBO.printEmployees(list);

}

}

**PasswordValidation**

class UserMainCode{

public static boolean checkPassword(String password) {

if(password.matches("^(?=.\*\\d)(?=.\*[a-z])(?=.\*[A-Z]).{8,}$"))

return true;

return false;

}

}

public class Main {

public static void main(String[] args) {

java.util.Scanner kb = new java.util.Scanner(System.in);

System.out.println(UserMainCode.checkPassword(kb.next()));

kb.close();

}

}