IRC_SKCT_Java2_CY_COD_Collection

Test Summary

- No. of Sections: 1
- No. of Questions: 10
- Total Duration: 120 min

Section 1 - Coding

Section Summary

- No. of Questions: 10
- Duration: 120 min

Additional Instructions:

None

Q1. **Equals method - Collections**

Let's dive deep into Set and explore its inbuilt functions. We usually perform equals operations to compare objects. Now try the same feature here.
Experiment with the equals() method in this problem.
Create Main class. Obtain two sets of numbers and check whether they are the same or not.
Note:
Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names, and method names should be the same as specified in the problem statement.

Input Format

The first line of the input consists of the value of n that represents the number of elements in both sets.
The next input is the elements of the first set separated by a space.
The third input is the elements of the second set separated by a space.

Output Format

The output prints whether the sets are equal or not.

Sample Input

```
2
10 20
10 20
```

Sample Output

```
Both sets are same
```

Sample Input

```
3
10 20 30
40 50 60
```

Sample Output

```
Both sets are different
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q2. Input 2 lines of text with multiple words separated by spaces. Using HashSet as an implementation of Set, print the set of words found in both lines.

Input Format

Input 2 lines with some words in common.

Output Format

The set of common words.

Sample Input

```
one two three
two four
```

Sample Output

```
[two]
```

Sample Input

```
first second third
third fourth first
```

Sample Output

```
[third, first]
```

Sample Input

```
hello hello
hello hello hello hello
```

Sample Output

```
[hello]
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q3. **HashSet**
Write a program to check if an element exists in the HashSet.

Input Format

The first line of the input consists of the number of inputs to be given
The next input is the elements in the Hash set.
The last input is the element to be searched.

Output Format

Output prints whether the element is present in the HashSet or not

Sample Input

```
3
c
c++
java
```

Sample Output

```
c is in the hash set.
```

Sample Input

```
3
c
c++
java
```

Sample Output

```
perl is not in the hash set.
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q4. Input a string with possibly repeated words separated by spaces. Print a count of unique words in the string using HashSet as an implementation of Set.

Input Format

Input a line with repeated words.

Output Format

The count of unique words

Sample Input

```
A name is a name only if it sounds like a name
```

Sample Output

```
9
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q5. A Student class is given with the following properties with appropriate getters, setters and toString overridden method
Roll No
Name
Address
Rank

Now the list of menu options are available for access. You need to write the following custom comparators
sortByName
sortByRollno
sortByRank

For the sortByRank comparator, implement the option for choosing the ascending and descending order for the rank.

sortByRank (boolean);
the Boolean value denotes whether the comparator should sort the options in ascending or descending order

The user should provide either ASC / DES (or) asc / des as the option for the sortByName comparator

Note : write the logic only for the comparators, all the logic are already written.

Sample Input

```
1
1,Gokul,Address1,2
1
2,Anima,Address2,5
```

Sample Output

```
Student Interactive Console :
1). Add User
2). Sort Student List by Roll no
```

Sample Input

```
1
1,Gokul,Address1,4
6
5
```

Sample Output

```
Student Interactive Console :
1). Add User
2). Sort Student List by Roll no
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q6. Create a class named **Book** with the following **private** attributes.

Include getters and setters.
Include default and parameterized constructor.
The format for the parameterized constructor is
Public Book(int id, String name, String author, String publisher, int quantity)
Create a **Main** class and in the main method get the details of the books and store them in the linked hash set.

Display all the book details and search for a book name in the set.

Input Format

The first line of the input consists of the number of books.
The next input is the book details.
The last input is the book name to be searched.

Output Format

The output prints the book details.
whether the element is present or not. Refer sample input and output.

Sample Input

```
2
1
let us c
yashwanth
```

Sample Output

```
1 let us c yashwanth BPB 8
2 operating systems galvin wiley 6
operating systems is present in the set
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q7. Use **remove()** and **isEmpty()** methods of the ArrayList API and implement them in our application. let's experiment with Hall class for performing these methods.
Create a class named **Hall** having the following private attributes.
Include getters and Setters for the attributes
Include default and parameterized constructors.
Format for a parameterized constructor is
Hall(String name, String contactNumber, Double costPerDay,String ownerName)
The Hall class contains the following method.

Create a driver class called **Main**. In the Main method, obtain input from the console If the list is empty and a remove action is performed display "**The list is empty**" and terminate. Display the Hall details by iterating the Hall List and calling the display() method after removing action.
Hint: Use isEmpty() and remove() methods of ArrayList api.
Use **System.out.printf("%-20s%-20s%-20s%-20s")** for displaying the Hall details in tabular form.
Note: Strictly adhere to the Object-Oriented specifications given in the problem statement.
All class names, attribute names, and method names should be the same as specified in the problem statement.

Input Format

The first line of the input consists of n.
The next input is the hall details.
The next input is the index of the element to be deleted.

Output Format

The output displays the hall details in tabular format after removing the element.

Sample Input

```
3
RR hall
9854785654
455
```

Sample Output

```
Name                Contact Number      CostperDay          Ow
RR hall             9854785654          455.0               Ra
PP hall             9632541578          357.0               Pr
```

Sample Input

```
0
2
```

Sample Output

```
The list is empty
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q8. Write a program to collect and store all the cards to assist the users in finding all the cards in a given symbol. This cards game consists of N number of cards. Get N number of cards details from the user and store the values in the Card object with the attributes symbol and number.

Store all the cards in a map with the symbol as its key and a list of cards as its value. The map is used here to easily group all the cards based on their symbol.

Once all the details are captured print all the distinct symbols in alphabetical order from the Map. For each symbol print all the card details, number of cards, and their sum respectively.

Input Format

Input number of cards, N >0
Input N card details as
Card Symbol
Card Number

Output Format

Distinct Symbols
For each Symbol
Cards in that Symbol
Number of Cards in that Symbol
Sum of numbers for that Symbol

Sample Input

Sample Output

```
5
b
2
```

```
Distinct Symbols are:
a b c d
Cards in a Symbol:
a 5
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q9. Playing cards during travel is a fun-filled experience. For this game, they wanted to collect all four unique symbols. Can you help these guys to collect unique symbols from a set of cards?
Create Card class with attributes symbol and number. From our main method collect each card details (symbol and number) from the user. Collect all these cards in a set, since the set is used to store unique values or objects. Cards need to be compared with each other to identify whether both the cards are the same symbol. For this, we need to implement equals method to tell whether both the cards are same or not.
Once we collect all four different symbols display the first occurrence of card details in alphabetical order.

Input Format

Input a card symbol
Input a card number
Repeat

Output Format

Four symbols gathered in *N* cards
Cards in set

Sample Input

```
a
2
b
1
```

Sample Output

```
Four symbols gathered in 5 cards
Cards in Set are:
a 2
b 1
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q10. Create a class **Student** with the following private attributes.

1. rno, mark1, mark2,mark3 as integer
2. name as String
3. avg as Double

Included appropriate getters/setters, also include constructors. And override the toString method to print details of the Student object. Include a method to calculate the average.
Create a main class **ArrayListSort** and get roll number, name, and marks of 3 subjects from the user using ArrayList and calculate average. Sort the students based on average and display.
Format: "%-20s%-20s%-20s".

Input Format

Number of students(N) in first-line as integer
Roll Number, Name, Marks of 3 subjects for N students as shown in sample input

Output Format

Student's average as shown in the sample output.

Sample Input

```
3
101
Ram
57
```

Sample Output

```
Roll No              Name                 Average
103                  Lohit                48.0
101                  Ram                  74.0
102                  Kanthick             91.0
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Answer Key & Solution

Section 1 - Coding

Q1

Test Case

Input

```
2
10 20
10 20
```

Output

```
Both sets are same
```

Weightage - 20

Input

```
3
10 20 30
40 50 60
```

Output

```
Both sets are different
```

Weightage - 20

Input

```
4
10 20 30 70
40 50 60 50
```

Output

```
Both sets are different
```

Weightage - 20

Input

```
4
10 20 30 60
10 20 60 30
```

Output

```
Both sets are same
```

Weightage - 20

Input

```
5
100 150 200 30 70
70 200 30 100 150
```

Output

```
Both sets are same
```

Weightage - 20

Sample Input

```
2
10 20
10 20
```

Sample Output

```
Both sets are same
```

Sample Input

```
3
10 20 30
40 50 60
```

Sample Output

```
Both sets are different
```

Solution

```
import java.io.*;
import java.util.*;
class Main {
public static void main(String [] args) {
    int i,n;
    Scanner sc = new Scanner(System.in);
    n = Integer.parseInt(sc.nextLine());
    Set<Integer> s1 = new HashSet<Integer>(n);
    Set<Integer> s2 = new HashSet<Integer>(n);
```

```
    for(i=0;i<n;i++) {
        s1.add(sc.nextInt());
    }
    for(i=0;i<n;i++) {
        s2.add(sc.nextInt());
    }
    if(s1.equals(s2)) {
        System.out.println("Both sets are same");
    }
    else {
        System.out.println("Both sets are different");
    }
}
}
```

Q2

Test Case

Input

```
first second third
third fourth first
```

Output

```
[third, first]
```

Weightage - 50

Input

```
order order
order order order
```

Output

```
[order]
```

Weightage - 50

Sample Input

```
one two three
two four
```

Sample Output

```
[two]
```

Sample Input

```
first second third
third fourth first
```

Sample Output

```
[third, first]
```

Sample Input

```
hello hello
hello hello hello hello
```

Sample Output

```
[hello]
```

Solution

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

class Q02Medium_Set
{
    public static void main (String[] args) throws java.lang.Exception
    {
        Scanner input = new Scanner(System. in);
        String line1 = input.nextLine();
        String line2 = input.nextLine();
        String words1[] = line1.split("[\\s]+");
        String words2[] = line2.split("[\\s]+");
        Set<String> uniqueWords1 = new HashSet<String>();
        uniqueWords1.addAll(Arrays.asList(words1));
        Set<String> uniqueWords2 = new HashSet<String>();
        uniqueWords2.addAll(Arrays.asList(words2));
```

```
        uniqueWords1.retainAll(uniqueWords2);
        System.out.println(uniqueWords1);
    }
}
```

Q3

Test Case

Input

```
3
c
c++
java
```

Output

```
c is in the hash set.
```

Weightage - 10

Input

```
3
c
c++
java
```

Output

```
php is not in the hash set.
```

Weightage - 15

Input

```
4
America
Australia
India
```

Output

```
India is in the hash set.
```

Weightage - 15

Input

```
4
America
Australia
India
```

Output

```
Italy is not in the hash set.
```

Weightage - 20

Input

```
5
America
Australia
India
```

Output

```
Maldives is not in the hash set.
```

Weightage - 20

Input

```
6
Cake
Cookies
Chocolates
```

Output

```
Chocolates is in the hash set.
```

Weightage - 20

Sample Input

```
3
c
c++
java
```

Sample Output

```
c is in the hash set.
```

Sample Input

```
3
c
c++
java
```

Sample Output

```
perl is not in the hash set.
```

Solution

```
import java.io.*;
import java.util.*;
class Main {
        public static void main(String[] args) {
            int i,n;
            Scanner sc = new Scanner(System.in);
            n = Integer.parseInt(sc.nextLine());
            Set<String> hs = new HashSet<>(n);
            for(i=0;i<n;i++) {
                hs.add(sc.nextLine());
            }
            String hsName = sc.nextLine();

            if(hs.contains(hsName)) {
                System.out.println(hsName + " is in the hash set.");
            } else {
                System.out.println(hsName + " is not in the hash set.");
            }
        }
    }
```

Q4

Test Case

Input

```
a quick brown fox jumps over the lazy dog
```

Output

```
9
```

Weightage - 10

Input

```
no words repeated
```

Output

```
3
```

Weightage - 10

Input

```
New Delhi is an urban district of Delhi which serves as t
```

Output

```
20
```

Weightage - 10

Input

```
Moisture that is lifted or otherwise forced to rise over
```

Output

```
33
```

Weightage - 15

Input

```
The Tortoise and the Hare is one of Aesops Fables and is
```

Output

```
43
```

Weightage - 15

Input

```
the reason that this version of the race is not widely kn
```

Output

```
63
```

Weightage - 20

Input

```
Outside of book production, there is an early 17th-century
```

Output

```
116
```

Weightage - 20

Sample Input

```
A name is a name only if it sounds like a name
```

Sample Output

```
9
```

Solution

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

class Q01Medium_Set
{
    public static void main (String[] args) throws java.lang.Exception
    {
        Scanner input = new Scanner(System. in);
        String line = input.nextLine();
        String words[] = line.split("[\\s]+");
        Set<String> strings = new HashSet<String>();
        for (String word: words) {
            strings.add(word);
        }
        System.out.println(strings.size());
    }
}
```

Q5

Test Case

Input

```
1
1,Hari,Address1,3
1
2,Ravi,Address2,5
```

Output

```
Student Interactive Console :
1). Add User
2). Sort Student List by Roll no
```

Weightage - 50

Input

```
1
1,Gokul,Address1,3
3
5
```

Output

```
Student Interactive Console :
1). Add User
2). Sort Student List by Roll no
```

Weightage - 25

Input

```
2
1
1,Ram,Address2,5
5
```

Output

```
Student Interactive Console :
1). Add User
2). Sort Student List by Roll no
```

Weightage - 25

Sample Input

```
1
1,Gokul,Address1,2
1
2,Anima,Address2,5
```

Sample Output

```
Student Interactive Console :
1). Add User
2). Sort Student List by Roll no
```

Sample Input

```
1
1,Gokul,Address1,4
6
5
```

Sample Output

```
Student Interactive Console :
1). Add User
2). Sort Student List by Roll no
```

Solution

Header

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

class Student
{
    int rollno, rank;
    String name, address;

    public Student(int rollno, String name, String address, int rank)
    {
        this.rollno = rollno;
        this.name = name;
        this.address = address;
        this.rank = rank;
    }

    public String toString()
    {
        return this.rollno + " " + this.name + " " + this.address + " " + this.rank;
    }
}



// Write the code below

class Sortbyroll implements Comparator<Student>
{

    public int compare(Student a, Student b)
    {
        return a.rollno - b.rollno;
    }
}

class Sortbyname implements Comparator<Student>
{

    public int compare(Student a, Student b)
    {
        return a.name.compareTo(b.name);
    }
}

class SortbyRank implements Comparator<Student>
{

    private boolean ascending;

    public SortbyRank(boolean ascending) {
```

```
        this.ascending = ascending;
    }

    public int compare(Student a, Student b)
    {
        return ascending ? a.rank - (b.rank) : b.rank - (a.rank);
    }
}
```

Footer

```
class Main
{
    public static void main (String[] args)
    {
        ArrayList<Student> ar = new ArrayList<Student>();

        Scanner s = new Scanner(System.in);

        String Welcome = "Student Interactive Console : \n1). Add User\n2). Sort Student List by Roll no\n3). Sort Student List by Name"
                + "\n4). sort Students by Rank\n5). Exit from System\n";
        int choice = 0;

do {

    System.out.println(Welcome);
    System.out.println("Enter your choice : ");
    choice = s.nextInt();


    switch(choice) {
        case 1 :
            s.nextLine();
            System.out.println("Enter the rollno, name, address and rank (separated by comma) ");
            String[] temp = s.nextLine().split(",");
            Student stu = new Student(Integer.parseInt(temp[0]), temp[1], temp[2], Integer.parseInt(temp[3]));
            ar.add(stu);
            break;
        case 2 :
            s.nextLine();
            Collections.sort(ar, new Sortbyroll());
            System.out.println("\nStudents List sorted by rollno");
            for (int i=0; i<ar.size(); i++)
                System.out.println(ar.get(i));
            break;
        case 3:
            s.nextLine();
            Collections.sort(ar, new Sortbyname());
            System.out.println("\nStudents List sorted by rollno");
            for (int i=0; i<ar.size(); i++)
                System.out.println(ar.get(i));
            break;
        case 4:
            s.nextLine();
            System.out.println("Sort by ascending or descending ( asc / des) ");
            boolean asc = s.nextLine().equalsIgnoreCase("asc") ? true : false;

            Collections.sort(ar, new SortbyRank(asc));
            System.out.println("\nStudents List sorted by Rank");
            for (int i=0; i<ar.size(); i++)
                System.out.println(ar.get(i));
            break;
        case 5 :
            System.out.println("Exiting ....");
            System.exit(0);
        default :
            s.nextLine();
            System.out.println("\nInvalid Input Try again !!!\n");

    }

} while (choice != 5 );
```

```
    }
}
```

Q6

Test Case

Input

```
2
1
let us c
yashwanth
```

Output

```
1 let us c yashwanth BPB 8
2 operating systems galvin wiley 6
operating systems is present in the set
```

Weightage - 20

Input

```
2
1
let us c
yashwanth
```

Output

```
1 let us c yashwanth BPB 8
2 operating systems galvin wiley 6
data communication is not present in the set
```

Weightage - 20

Input

```
3
1
let us c
yashwanth
```

Output

```
1 let us c yashwanth BPB 8
2 operating systems galvin wiley 6
3 data communication forouzan mc graw hill 5
let us c is present in the set
```

Weightage - 20

Input

```
3
1
let us c
yashwanth
```

Output

```
1 let us c yashwanth BPB 8
2 operating systems galvin wiley 6
3 data communication forouzan mc graw hill 5
computer networks is not present in the set
```

Weightage - 20

Input

```
4
1
let us c
yashwanth
```

Output

```
1 let us c yashwanth BPB 8
2 operating systems galvin wiley 6
3 data communication forouzan mc graw hill 5
4 computer networks yashwanth BPB 4
```

Weightage - 20

Sample Input

```
2
1
let us c
yashwanth
```

Sample Output

```
1 let us c yashwanth BPB 8
2 operating systems galvin wiley 6
operating systems is present in the set
```

Solution

```
import java.io.*;
import java.util.*;
class Book {
    private int id;
    private String name;
    private String author;
    private String publisher;
    private int quantity;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
```

```
public String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
}
public String getAuthor() {
    return author;
}
public void setAuthor(String author) {
    this.author = author;
}
public String getPublisher() {
    return publisher;
}
public void setPublisher(String publisher) {
    this.publisher = publisher;
}
public int getQuantity() {
    return quantity;
}
public void setQuantity(int quantity) {
    this.quantity = quantity;
}
public Book() {
        this.id = 0;
        this.name = null;
        this.author = null;
        this.publisher = null;
        this.quantity = 0;
    }
    public Book(int id, String name, String author, String publisher,int quantity) {
        this.id = id;
        this.name = name;
        this.author = author;
        this.publisher = publisher;
        this.quantity = quantity;
    }
}
class Main {
    public static void main(String [] args) {
        int i,n,flag = 0;
        Scanner sc = new Scanner(System.in);
        n = Integer.parseInt(sc.nextLine());
        Book [] b = new Book[n];
        LinkedHashSet<Book> bb = new LinkedHashSet<Book>(n);
        for(i=0;i<n;i++) {
            b[i] = new Book();
            b[i].setId(Integer.parseInt(sc.nextLine()));
            b[i].setName(sc.nextLine());
            b[i].setAuthor(sc.nextLine());
            b[i].setPublisher(sc.nextLine());
            b[i].setQuantity(Integer.parseInt(sc.nextLine()));
            bb.add(b[i]);
        }
        String searchBook = sc.nextLine();
        for(Book b1 : bb) {
            System.out.println(b1.getId()+" "+b1.getName()+" "+b1.getAuthor()+" "+b1.getPublisher()+" "+b1.getQuantity());
        }
        for(Book b2 : bb) {
            if(b2.getName().contains(searchBook)) {
                flag = 1;
                break;
            }
        }
        if(flag == 1) {
            System.out.println(searchBook+" is present in the set");
        }
        else {
            System.out.println(searchBook+" is not present in the set");
        }
    }
}
```

Q7

Test Case

Input

```
3
RR hall
9854785654
455
```

Output

```
Name                    Contact Number        CostperDay
RR hall                 9854785654            455.0
PP hall                 9632541578            357.0
```

Weightage - 20

Input

```
4
RR hall
9854785654
455
```

Output

```
Name                    Contact Number        CostperDay
RR hall                 9854785654            455.0
KK hall                 95478563221           258.0
```

Weightage - 20

Input

```
5
RR hall
9854785654
455
```

Output

```
Name                    Contact Number        CostperDay
RR hall                 9854785654            455.0
KK hall                 95478563221           258.0
```

Weightage - 20

Input

```
6
RR hall
9854785654
455
```

Output

```
Name                    Contact Number        CostperDay
RR hall                 9854785654            455.0
KK hall                 95478563221           258.0
```

Weightage - 20

Input

```
0
2
```

Output

```
The list is empty
```

Weightage - 20

Sample Input

```
3
RR hall
9854785654
455
```

Sample Output

```
Name                    Contact Number        CostperDay
RR hall                 9854785654            455.0
PP hall                 9632541578            357.0
```

Sample Input

```
0
2
```

Sample Output

```
The list is empty
```

Solution

```
import java.io.*;
import java.util.*;
class Hall {
    private String name;
    private String contactNumber;
    private double costPerDay;
    private String ownerName;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
```

```
}
public String getContactNumber() {
    return contactNumber;
}
public void setContactNumber(String contactNumber) {
    this.contactNumber = contactNumber;
}
public double getCostPerDay() {
    return costPerDay;
}
public void setCostPerDay(double costPerDay) {
    this.costPerDay = costPerDay;
}
public String getOwnerName() {
    return ownerName;
}
public void setOwnerName(String ownerName) {
    this.ownerName = ownerName;
}
public Hall() {
    this.name = null;
    this.contactNumber = null;
    this.costPerDay =0;
    this.ownerName = null;
}
    public Hall(String name, String contactNumber, Double costPerDay,String ownerName) {
        this.name = name;
        this.contactNumber = contactNumber;
        this.costPerDay = costPerDay;
        this.ownerName = ownerName;
    }
    public void display() {
        System.out.printf("%-20s%-20s%-20s%-20s\n",name,contactNumber,costPerDay,ownerName);
    }
}
class Main {
    public static void main(String [] args) {
        int i,n;
        Scanner sc = new Scanner(System.in);
        n = Integer.parseInt(sc.nextLine());
        ArrayList<Hall> h1 = new ArrayList<>(n);
        Hall [] h = new Hall[n];
        for(i=0;i<n;i++) {
            h[i] = new Hall();
            h[i].setName(sc.nextLine());
            h[i].setContactNumber(sc.nextLine());
            h[i].setCostPerDay(Double.parseDouble(sc.nextLine()));
            h[i].setOwnerName(sc.nextLine());
            h1.add(h[i]);
        }
        int index = Integer.parseInt(sc.nextLine());
        if(h1.isEmpty()) {
            System.out.println("The list is empty");
        }
        else {
        h1.remove(index);
        System.out.printf("%-20s%-20s%-20s%-20s\n","Name","Contact Number","CostperDay","Owner Name");
        }

        for(i=0;i<n-1;i++) {
            h1.get(i).display();
        }
    }
}
```

Q8

Test Case

Input	Output
6 a 1 b	Distinct Symbols are: a b c d Cards in a Symbol: a 1

Weightage - 100

Sample Input

```
5
b
2
```

Sample Output

```
Distinct Symbols are:
a b c d
Cards in a Symbol:
```

Solution

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


class Q04Easy_Collections2
{
    static class Card {
        private String symbol;
        private Integer number;

        public Card(String symbol, Integer num) {
            this.symbol = symbol;
            this.number = num;
        }
        public String getSymbol() {
            return symbol;
        }
        public void setSymbol(String symbol) {
            this.symbol = symbol;
        }
        public Integer getNumber() {
            return number;
        }
        public void setNumber(int number) {
            this.number = number;
        }
        public String toString(){
            return this.symbol +" " + this.number;
        }
}

static class CardComp implements Comparator<Card>{

    @Override
    public int compare(Card e1, Card e2) {
        int cmp = e1.getSymbol().compareTo(e2.getSymbol());
        if (cmp == 0) return e1.getNumber() - e2.getNumber();
        return cmp;
    }
}

public static void main (String[] args) throws java.lang.Exception
{
    Scanner input = new Scanner(System. in);
    boolean err = false;
    int count = 0;
    TreeMap<String, TreeSet<Card>> cards = new TreeMap<String, TreeSet<Card>>();
    int cardCount = 0;
    do {
        try {
            cardCount = Integer.parseInt(input. nextLine());
            if (cardCount > 0) err = false;
            else err = true;
        } catch (NumberFormatException e) {
            err = true;
        }
    } while (err);

    while (cardCount > 0) {
        // Input card
        String symbol = input.nextLine();
        Integer value = null;
```

```
    do {
        try {
            value = Integer.parseInt(input.nextLine());
            err = false;
        } catch (NumberFormatException e) {
            System.out.println("Invalid Input");
            err = true;
        }
    } while (err);
    if (!cards.containsKey(symbol)) {
        cards.put(symbol, new TreeSet<Card>(new CardComp()));
    }
    cards.get(symbol).add(new Card(symbol, value));
    count++;
    cardCount--;
}

System.out.println("Distinct Symbols are: ");
for (String key: cards.keySet()) {
    System.out.print(key + " ");
}
System.out.println();
for (String key: cards.keySet()) {
    System.out.println("Cards in " + key + " Symbol:");
    TreeSet<Card> value = cards.get(key);
            int sum = 0;
            for (Card card: value) {
                System.out.println(card);
                sum += card.getNumber();
            }
            System.out.println("Number of Cards: " + value.size());
            System.out.println("Sum of Numbers: " + sum);

        }
    }
}
```

Q9

Test Case

Input

```
x
5
b
1
```

Output

```
Four symbols gathered in 5 cards
Cards in Set are:
b 1
c 3
```

Weightage - 10

Input

```
a
2
d
4
```

Output

```
Four symbols gathered in 9 cards
Cards in Set are:
a 2
d 4
```

Weightage - 15

Input

```
j
5
j
9
```

Output

```
Four symbols gathered in 7 cards
Cards in Set are:
g 6
j 5
```

Weightage - 15

Input

```
g
8
q
1
```

Output

```
Four symbols gathered in 4 cards
Cards in Set are:
g 8
j 1
```

Weightage - 15

Input

```
e
4
r
4
```

Output

```
Four symbols gathered in 6 cards
Cards in Set are:
e 4
g 9
```

Weightage - 15

Input

```
p
5
i
6
```

Output

```
Four symbols gathered in 8 cards
Cards in Set are:
i 6
p 5
```

Weightage - 15

Input

```
l
4
p
5
```

Output

```
Four symbols gathered in 4 cards
Cards in Set are:
f 7
l 4
```

Weightage - 15

Sample Input

```
a
2
b
1
```

Sample Output

```
Four symbols gathered in 5 cards
Cards in Set are:
a 2
b 1
```

Solution

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


class Q03Easy_Collections1
{
    static class Card {
        private String symbol;
        private Integer number;

        public Card(String symbol, Integer num) {
            this.symbol = symbol;
            this.number = num;
        }
        public String getSymbol() {
            return symbol;
        }
        public void setSymbol(String symbol) {
            this.symbol = symbol;
        }
        public Integer getNumber() {
            return number;
        }
        public void setNumber(int number) {
            this.number = number;
        }
        public String toString(){
            return this.symbol +" " + this.number;
        }
}

static class CardComp implements Comparator<Card>{

    @Override
    public int compare(Card e1, Card e2) {
```

```
        return e1.getSymbol().compareTo(e2.getSymbol());
    }
}

public static void main (String[] args) throws java.lang.Exception
{
    Scanner input = new Scanner(System. in);
    boolean err = false;
    int count = 0;
    TreeSet<Card> cards = new TreeSet<Card>(new CardComp());
    while (cards.size() < 4) {
        // Input card
        String symbol = input.nextLine();
        Integer value = null;
        do {
            try {
                value = Integer.parseInt(input.nextLine());
                err = false;
            } catch (NumberFormatException e) {
                System.out.println("Invalid Input");
                err = true;
            }
        } while (err);
        cards.add(new Card(symbol, value));
        count++;
        }
        System.out.println("Four symbols gathered in " + count + " cards");
        System.out.println("Cards in Set are: ");

        for (Card card: cards) {
            System.out.println(card);
        }
    }
}
```

Q10

Test Case

Input

```
3
101
Ramu
57
```

Output

```
Roll No                 Name                    Average
103                      Lohit                   28.0
101                      Ramu                    74.0
```

Weightage - 25

Input

```
5
102
Karthick
58
```

Output

```
Roll No                 Name                    Average
105                      Kumar                   48.0
103                      Lohit                   51.0
```

Weightage - 25

Input

```
4
1002
Karthick
58
```

Output

```
Roll No                 Name                    Average
1002                     Karthick               61.0
1003                     Lohit                   68.0
```

Weightage - 25

Input

```
3
1001
Ramesh
27
```

Output

```
Roll No                 Name                    Average
1001                     Ramesh                  54.0
1003                     Lokesh                  88.0
```

Weightage - 25

Sample Input

```
3
101
Ram
57
```

Sample Output

```
Roll No              Name                 Average
103                   Lohit                48.0
101                   Ram                  74.0
```

Solution

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

class Student implements Comparable<Student>{
    private int rno;
    private String name;
    private int m1,m2,m3;
    private double avg;

    public Student(int rno, String name, int m1, int m2, int m3) {
        super();
        this.rno = rno;
        this.name = name;
        this.m1 = m1;
        this.m2 = m2;
        this.m3 = m3;
    }

    private double calculateAvg() {
        double avg =(m1+m2+m3)/3;
        return avg;
    }

    @Override
    public String toString() {
        return String.format("%-20s%-20s%-20s", rno , name , calculateAvg()) ;
    }

    @Override
    public int compareTo(Student obj) {
        return calculateAvg()>obj.calculateAvg()?1:-1;
    }

}


 class ArrayListSort{
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());
        int rno,m1,m2,m3;
        String name;
        List<Student> stud = new ArrayList<>();
        for(int i=0;i<n;i++){
            rno =Integer.parseInt(sc.nextLine());
            name =sc.nextLine();
            m1 =Integer.parseInt(sc.nextLine());
            m2 =Integer.parseInt(sc.nextLine());
            m3 =Integer.parseInt(sc.nextLine());
            stud.add(new Student(rno,name,m1,m2,m3));
        }

        Collections.sort(stud);
        System.out.printf("%-20s%-20s%-20s\n", "Roll No" , "Name" , "Average") ;

        for (Student s : stud) {
            System.out.println(s);
        }
    }
}
```