11. Concatenate Characters

Given an array of Strings, write a program to take the last character of each string and make a new String by concatenating it.

Include a class **UserMainCode** with a static method **“concatCharacter”** that accepts a String array as input and returns the new String.

Create a class **Main** which would get the String array as input and call the static method **concatCharacter** present in the UserMainCode.

Input and Output Format:
The first line of the input consists of an integer n that corresponds to the number of strings in the input string array.
The next n lines of the input consist of the strings in the input string array.
Output consists of a string.

Sample Input:

```
3
ab
a
abcd
```

Sample Output:

```
bad
```

Solution:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner s = newScanner(System.in);
int n = s.nextInt();
String[] str = new String[n];
for (int i = 0; i < n; i++)
str[i] = s.next();
System.out.println(User.concatCharacter(str));
}
}
```

```
publicclass User {
publicstatic String concatCharacter(String[] s) {

StringBuffer sb = newStringBuffer();
for (int i = 0; i < s.length; i++) {
sb.append(s[i].charAt(s[i].length() - 1));
}
return sb.toString();
}
}
```

12. Anagram

Write a program to check whether the two given strings are anagrams.

Note: Rearranging the letters of a word or phrase to produce a new word or phrase, using all the original letters exactly once is called Anagram."

Include a class **UserMainCode** with a static method **"getAnagram"** that accepts 2 strings as arguments and returns an int. The method returns 1 if the 2 strings are anagrams. Else it returns -1.

Create a class **Main** which would get 2 Strings as input and call the static method **getAnagram** present in the UserMainCode.

Input and Output Format:
Input consists of 2 strings. Assume that all characters in the string are lower case letters.
Output consists of a string that is either "Anagrams" or "Not Anagrams".

Sample Input 1:
eleven plus two
twelve plus one

Sample Output 1:
Anagrams

Sample Input 2:
orchestra
carthorse

Sample Output 2:
Anagrams

Sample Input 3:
cognizant
technologies

Sample Output 3:
Not Anagrams

Solutions:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner s = newScanner(System.in);
String s1 = s.nextLine();
String s2 = s.nextLine();
int result = User.getAnagrams(s1, s2);
if (result == 1)
System.out.println("Anagrams");
else
System.out.println("Not Anagrams");
}
}


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

publicclass User {
publicstaticint getAnagrams(String s1, String s2) {

String str1 = s1.toLowerCase();
String str2 = s2.toLowerCase();
ArrayList<Character> al1 = new ArrayList<Character>();
ArrayList<Character> al2 = new ArrayList<Character>();
ArrayList<Character> al3 = new ArrayList<Character>();
int res = 0;
for (int i = 0; i < s1.length(); i++)
al1.add(str1.charAt(i));
for (int i = 0; i < s2.length(); i++)
al2.add(str2.charAt(i));
al3.add(' ');
al1.removeAll(al3);
al2.removeAll(al3);
Collections.sort(al1);
Collections.sort(al2);
if (al1.equals(al2))
res = 1;
else
res = -1;
return res;
}
}


public static void main(String[] args) {

            Scanner sc = new Scanner(System.in);
            String s1 = sc.nextLine();
            String s2 = sc.nextLine();
            boolean b =Anagrams.check(s1, s2);
            if (b == true)
            System.out.println("TRUE");
            else
            System.out.println("FALSE");

      }

}




      public class Anagrams
      {
            public static boolean check(String s1,String s2)
            {
            boolean res=false;
                  ArrayList<Character> a1=new ArrayList<Character>();
                  ArrayList<Character> a2=new ArrayList<Character>();
                  for(int i=0;i<s1.length();i++)
                  {
                        a1.add(s1.charAt(i));
                  }

                  for(int i=0;i<s2.length();i++)
                  {
                        a2.add(s2.charAt(i));
                  }
                  Collections.sort(a1);
                  Collections.sort(a2);


                  if((a1.containsAll(a2))|| (a2.containsAll(a1)))
                  {
                        res=true;
                  }
                  return res;
            }
      }
```

13. Calculate Meter Reading

Given 2 strings corresponding to the previous meter reading and the current meter reading, write a program to calculate electricity bill.
The input string is in the format ""AAAAAXXXXX"".
AAAAA is the meter code and XXXXX is the meter reading.
FORMULA: (XXXXX-XXXXX)*4

Hint: if AAAAA of input1 and input2 are equal then separate the XXXXX from string and convert to integer. Assume that AAAAA of the 2 input strings will always be equal.

Include a class **UserMainCode** with a static method **“calculateMeterReading”** that accepts 2 String arguments and returns an integer that corresponds to the electricity bill. The 1st argument corresponds to the previous meter reading and the 2ndarguement corresponds to the current meter reading.

Create a class **Main** which would get 2 Strings as input and call the static method **calculateMeterReading** present in the UserMainCode.

Input and Output Format:
Input consists of 2 strings. The first input corresponds to the previous meter reading and the second input corresponds to the current meter reading.

Output consists of an integer that corresponds to the electricity bill.

Sample Input:
CSECE12390
CSECE12400

Sample Output:
40

Solution:

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

```
publicclass Main {
publicstaticvoid main(String[] args) {
Scanner s = newScanner(System.in);
String s1 = s.nextLine();
String s2 = s.nextLine();
System.out.println(User.calculateMeterReading(s1, s2));
}
}
```

```
publicclass User {
publicstaticint calculateMeterReading(String s1, String s2) {
String str1 = s1.substring(s1.length() / 2);
String str2 = s2.substring(s2.length() / 2);
int a = Integer.parseInt(str1);
int b = Integer.parseInt(str2);
int res = (b - a) * 4;
return res;
}
}
```

14. Retirement

Given an input as HashMap which contains key as the ID and dob as value of employees, write a program to find out employees eligible for retirement. A person is eligible for retirement if his age is greater than or equal to 60.

Assume that the current date is 01/01/2014.

Include a class **UserMainCode** with a static method "retirementEmployeeList" that accepts a HashMap<String,String> as input and returns a ArrayList<String>. In this method, add the Employee IDs of all the retirement eligible persons to list and return the sorted list.
(Assume date is in dd/MM/yyyy format).

Create a class **Main** which would get the HashMap as input and call the static method **retirementEmployeeList** present in the UserMainCode.

Input and Output Format:
The first line of the input consists of an integer n, that corresponds to the number of employees.
The next 2 lines of the input consists of strings that correspond to the id and dob of employee 1.
The next 2 lines of the input consists of strings that correspond to the id and dob of employee 2.

and so on...
Output consists of the list of employee ids eligible for retirement in sorted order.

Sample Input :

```
4
C1010
02/11/1987
C2020
15/02/1980
C3030
14/12/1952
T4040
20/02/1950
```

Sample Output:

```
[C3030, T4040]
```

Solution:

```
import java.text.ParseException;
importjava.util.LinkedHashMap;
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) throws ParseException {
Scanner s = newScanner(System.in);
int n = s.nextInt();
LinkedHashMap<String, String> hm = newLinkedHashMap<String, String>();
for (int i = 0; i < n; i++)
hm.put(s.next(), s.next());
System.out.println(User.retirementEmployeeList(hm));
}
}

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.Iterator;
import java.util.LinkedHashMap;

publicclass User {
publicstaticArrayList<String> retirementEmployeeList(
LinkedHashMap<String, String> hm) throws ParseException {
```

```
ArrayList<String> al = new ArrayList<String>();
SimpleDateFormat sdf = newSimpleDateFormat("dd/MM/yyyy");
String s = "01/01/2014";
Date d2 = sdf.parse(s);
Date d1 = newDate();
Iterator<String> itr = hm.keySet().iterator();
while (itr.hasNext()) {
String key = itr.next();
String val = hm.get(key);
d1 = sdf.parse(val);
Calendar c = Calendar.getInstance();
c.setTime(d1);
int y1 = c.get(Calendar.YEAR);
int m1 = c.get(Calendar.MONTH);
int day1 = c.get(Calendar.DAY_OF_MONTH);
c.setTime(d2);
int y2 = c.get(Calendar.YEAR);
int m2 = c.get(Calendar.MONTH);
int day2 = c.get(Calendar.DAY_OF_MONTH);
int y = Math.abs(y1 - y2);
if (m1 == m2) {
if (day1 > day2)
y--;
} elseif (m1 > m2)
y--;
if (y >= 60)
al.add(key);
}
return al;
}
}
```

```
import java.text.ParseException;
import java.util.HashMap;
import java.util.Scanner;

public class NewClassMain {

      public static void main(String[] args) throws ParseException {


            Scanner sc=new Scanner(System.in);
            int n=sc.nextInt();
```

```
		int i=0;
		sc.nextLine();
		HashMap<String,String> hm=new HashMap<String,String>();
		for(i=0;i<n;i++)
		{
			hm.put(sc.nextLine(),sc.nextLine());
		}

		System.out.println(NewClass.retirement(hm));

			}

}



public class NewClass {

	public static ArrayList<String> retirement(HashMap<String,String> hm)
throws ParseException
	{
		ArrayList<String> a1=new ArrayList<String>();
	String s="01/01/2014";
	Iterator<String> itr=hm.keySet().iterator();
	while(itr.hasNext())
	{
	String k=itr.next();
	String dob=hm.get(k);

	SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
	Date d1=sdf.parse(dob);
	Date d2=sdf.parse(s);

	Calendar cal=Calendar.getInstance();
	cal.setTime(d1);

	int y1= cal.get(Calendar.YEAR);
		cal.setTime(d2);

	int y2= cal.get(Calendar.YEAR);

	int res=y2-y1;
	if(res>=60)

	{
		a1.add(k);
	}

	}

	Collections.sort(a1);

	return a1;

	}
```

```
}
```

15. Kaprekar Number

Write a program to check whether the given input number is a Kaprekar number or not.
Note : A positive whole number ‘n’ that has ‘d’ number of digits is squared and split into two pieces, a right-hand piece that has ‘d’ digits and a left-hand piece that has remaining ‘d’ or ‘d-1’ digits. If the sum of the two pieces is equal to the number, then ‘n’ is a Kaprekar number.

If its Kaprekar number assign to output variable 1 else -1.
Example 1:
Input1:9
9^2 = 81, right-hand piece of 81 = 1 and left hand piece of 81 = 8
Sum = 1 + 8 = 9, i.e. equal to the number. Hence, 9 is a Kaprekar number.

Example 2:
Input1:45
Hint:
45^2 = 2025, right-hand piece of 2025 = 25 and left hand piece of 2025 = 20
Sum = 25 + 20 = 45, i.e. equal to the number. Hence, 45 is a Kaprekar number."

Include a class **UserMainCode** with a static method “**getKaprekarNumber**” that accepts an integer argument and returns an integer. The method returns 1 if the input integer is a Kaprekar number. Else the method returns -1.

Create a class **Main** which would get the an Integer as input and call the static method **getKaprekarNumber** present in the UserMainCode.

Input and Output Format:
Input consists of an integer.
Output consists of a single string that is either "Kaprekar Number" or "Not A Kaprekar Number"

Sample Input 1:
9

Sample Output 1:
Kaprekar Number

Sample Input 2:
45

Sample Output 2:
Kaprekar Number

Sample Input 3:
4

Sample Output 3:
Not A Kaprekar Number

Solution:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
int n = sc.nextInt();
int i = User.getKaprekarNumber(n);
if (i == 1)
System.out.println("Kaprekar Number");
else
System.out.println("Not Kaprekar Number");
}
}


public class User {
public static int getKaprekarNumber(int temp) {
int n = temp;
int sq = n * n;
int sqr=sq;
int res = 0;
int count = 0;
while (sq != 0) {
count++;
sq= sq / 10;
}
//String a = Integer.toString(sqr);

String a=String.valueOf(sqr);


String n1 = a.substring(count/2);
String n2 = a.substring(0,count/2);
int i = Integer.parseInt(n1);
int j = Integer.parseInt(n2);
if ((i + j) == temp)
res = 1;
else
res = -1;
return res;
}
}
```

```
public class Palindrome {
      public static  int  removeDuplicate(int n)
      {
            int temp = n;
            int sq = n * n;
            int sqr=sq;
            int res = 0;
            String sqs=String.valueOf(sq);
            int len=sqs.length();
            String a = String.valueOf(sqr);
            String n1 = a.substring(len/2);
            String n2 = a.substring(0,len/2);
            int i = Integer.parseInt(n1);
            int j = Integer.parseInt(n2);
            if ((i + j) == temp)
            res = 1;
            else
            res = -1;
            return res;

}
}
```

16. Vowels

Given a String input, write a program to find the word which has the the maximum number of vowels. If two or more words have the maximum number of vowels, print the first word.

Include a class **UserMainCode** with a static method "**storeMaxVowelWord**" that accepts a string argument and returns the word containing the maximum number of vowels.

Create a class **Main** which would get the a String as input and call the static method **storeMaxVowelWord** present in the UserMainCode.

Input and Output Format:
Input consists of a string. The string may contain both lower case and upper case letters.
Output consists of a string.

Sample Input :
What is your name?

Sample Output :
Your

Solution:

```
import java.text.ParseException;
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) throws ParseException {
Scanner sc = newScanner(System.in);
String s = sc.nextLine();
System.out.println(User.storeMaxVowelWord(s));
}
}


import java.util.StringTokenizer;

publicclass User {
publicstatic String storeMaxVowelWord(String s) {
StringTokenizer st = new StringTokenizer(s, " ");
int count = 0, max = 0;
String s2 = null;
while (st.hasMoreTokens()) {
String s1 = st.nextToken();
count = 0;
for (int i = 0; i <s1.length(); i++) {
if (s1.charAt(i) == 'a' || s1.charAt(i) == 'e'
|| s1.charAt(i) == 'i' || s1.charAt(i) == 'o'
|| s1.charAt(i) == 'u' || s1.charAt(i) == 'A'
|| s1.charAt(i) == 'E' || s1.charAt(i) == 'I'
|| s1.charAt(i) == 'O' || s1.charAt(i) == 'U')
count++;
}
if (count > max) {
max = count;
s2 = s1;
}
}
return s2;

}
}
```

17. Unique Characters

Given a String as input , write a program to count and print the number of unique characters in it.

Include a class **UserMainCode** with a static method “**checkUnique**” that accepts a String as input and returns the number of unique characters in it. If there are no unique characters in the string, the method returns -1.

Create a class **Main** which would get a String as input and call the static method **checkUnique** present in the UserMainCode.

Input and Output Format:
Input consists of a string.
Output consists of an integer.

Sample Input 1:
HOWAREYOU

Sample Output 1:
7

(Hint :Unique characters are : H,W,A,R,E,Y,U and other characters are repeating)

Sample Input 2:
MAMA

Sample Output2:
-1

Solution:

```
import java.util.Scanner;

publicclass Main {

publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
String s = sc.next();
System.out.println(User.checkUnique(s));
}
}


publicclass User {
publicstaticint checkUnique(String s) {
StringBuffer sb = new StringBuffer(s);
int len = s.length();
int i = 0, j = 0, count;
for (i = 0; i < len; i++) {
count = 0;
for (j = i + 1; j < len; j++) {
if (sb.charAt(i) == sb.charAt(j)) {
sb.deleteCharAt(j);
count++;
j--;
len--;
}
}

if (count > 0) {
sb.deleteCharAt(i);
i--;
len--;
}
}
if(sb.length()==0)
        return -1;
else
return sb.length();
}
}
```

18. Average of Primes

Write a program to read an array and find average of all elements located at index i, where i is a prime number. Type cast the average to an int and return as output. The index starts from 0.

Include a class UserMainCode with a static method **addPrimeIndex** which accepts a single integer array. The return type (integer) should be the average of all elements located at index i where i is a prime number.

Create a Class Main which would be used to accept Input array and call the static method present in UserMainCode.

Input and Output Format:

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of a single Integer.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20 and minimum number of elements is 3.

Sample Input 1:

4

2

5

2

4

Sample Output 1:

3

Solutions:

```
import java.text.ParseException;
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) throws ParseException {
Scanner sc = newScanner(System.in);
int n = sc.nextInt();
int[] a = newint[20];
for (int i = 0; i < n; i++)
a[i] = sc.nextInt();
System.out.println(User.addPrimeIndex(a));
}
}


public class User {
public static int addPrimeIndex(int a[],int n) {
int count=0,sum=0,temp=0;
int avg=0;
for(int i=2;i<=n;i++)
{
count=0;
for(int j=1;j<i;j++)
{
if(i%j==0)
count++;

}
if(count==1)
{
temp++;
sum=sum+a[i];
}
}
avg=sum/temp;
return avg;

}
}
```

19. **ArrayList and Set Operations**

Write a program that performs the following actions:

1. Read 2n integers as input & a set operator (of type char).
2. Create two arraylists to store n elements in each arraylist.

3. Write a function **performSetOperations** which accepts these two arraylist and the set operator as input.
4. The function would perform the following set operations:.

   '+' for SET-UNION

   '*' for SET-INTERSECTION

   '-' for SET-DIFFERENCE

   Refer to sample inputs for more details.
5. Return the resultant arraylist.

Include a class UserMainCode with the static method **performSetOperations** which accepts two arraylist and returns an arraylist.

Create a Class Main which would be used to read 2n+1 integers and call the static method present in UserMainCode.

Note:

- The index of first element is 0.

Input and Output Format:

Input consists of 2n+2 integers. The first integer denotes the size of the arraylist, the next n integers are values to the first arraylist, and the next n integers are values to the second arraylist and the last input corresponds to that set operation type.

Output consists of a modified arraylist as per step 4.

Refer sample output for formatting specifications.

Sample Input 1:

3

1

2

3

3

5

7

+

Sample Output 1:

1

2

3

5

7

Sample Input 2:

4

10

9

8

7

2

4

6

8

*****

Sample Output 2:

8

Sample Input 3:

4

5

10

15

20

0

10

12

20

-

Sample Output 3:

5

15

Solution:

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

public class Main {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
ArrayList<Integer> al1 = new ArrayList<Integer>();
ArrayList<Integer> al2 = new ArrayList<Integer>();
ArrayList<Integer> res = new ArrayList<Integer>();
for (int i = 0; i < n; i++)
al1.add(sc.nextInt());
for (int i = 0; i < n; i++)
al2.add(sc.nextInt());
char c = sc.next().charAt(0);
res = User.performSetOperations(al1, al2, c);
for (int i = 0; i < res.size(); i++)
System.out.println(res.get(i));
}
}


import java.util.ArrayList;
import java.util.LinkedHashSet;

public class User {
public static ArrayList<Integer> performSetOperations(
ArrayList<Integer> al1, ArrayList<Integer> al2, char c) {

LinkedHashSet<Integer> h = new LinkedHashSet<Integer>();
ArrayList<Integer> al3 = new ArrayList<Integer>();
switch (c) {
```

```
case '+':
al1.addAll(al2);
h.addAll(al1);
al3.addAll(h);
break;
case '*':
for (int i = 0; i < al1.size(); i++) {
for (int j = 0; j < al2.size(); j++) {
if (al1.get(i) == al2.get(j)) {
al3.add(al1.get(i));
}
}
}
break;
case '-':
for (int i = 0; i < al1.size(); i++) {
for (int j = 0; j < al2.size(); j++) {
if (al1.get(i) == al2.get(j)) {
al1.remove(i);
}
}
}
al3.addAll(al1);
break;

}

return al3;
}
}
```

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Scanner;

class Main
{
    public static void main(String[] arg)
    {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        ArrayList<Integer> aa=new ArrayList<Integer>();
        for(int i=0;i<n;i++)
        {
            aa.add(sc.nextInt());

        }
        ArrayList<Integer> aa2=new ArrayList<Integer>();
        for(int i=0;i<n;i++)
        {
            aa2.add(sc.nextInt());

        }
        char c=sc.next().charAt(0);

        ArrayList<Integer> op=new ArrayList<Integer>();

        op=MainClass.setOPeration(n, aa, aa2, c);
        Iterator<Integer> itr=op.iterator();
        while(itr.hasNext())
        {
            int a=(Integer)itr.next();
            System.out.println(a);
        }


    }}




import java.util.ArrayList;


public class MainClass {


    public static ArrayList<Integer> setOPeration
    (int n,ArrayList<Integer>aa,ArrayList<Integer>aa2,char c)
    {

        ArrayList<Integer> aa3= new ArrayList<Integer>();
        if(c=='+')
        {
            aa.removeAll(aa2);
            aa.addAll(aa2);
            aa3=aa;
        }


        if(c=='*')
        {
            aa.retainAll(aa2);
            aa3=aa;

            }
    if(c=='-')
    {
        aa.removeAll(aa2);
        aa3=aa;
    }
return aa3;
}
}
```

20. **Largest Span**

Write a program to read an array and find the size of largest span in the given array ""span"" is the number of elements between two repeated numbers including both numbers. An array with single element has a span of 1.

.

Include a class UserMainCode with a static method **getMaxSpan** which accepts a single integer array. The return type (integer) should be the size of largest span.

Create a Class Main which would be used to accept Input array and call the static method present in UserMainCode.

Input and Output Format:

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of a single Integer.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

Sample Input 1:

5

1

2

1

1

3

Sample Output 1:

4

Sample Input 2:

7

1

4

2

1

4

1

5

Sample Output 2:

6

Solution:

```
import java.util.Scanner;

publicclass Main {

publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
int n = sc.nextInt();
int[] a = newint[n];
for (int i = 0; i < n; i++)
a[i] = sc.nextInt();
System.out.println(User.getLargestSpan(a));

}
}


publicclass User {
publicstaticint getLargestSpan(int[] a) {
int len = a.length;
int i = 0, j = 0, e = 0, count = 0;
for (i = 0; i < len; i++) {
for (j = i + 1; j < len; j++) {
if (a[i] == a[j]) {
e = j;
}
}
if (e - i > count)
count = e - i;
}
return count + 1;
}
}
```