**SET B CORE JAVA (CSL5303) ST3**

**1 mark MCQs:**

1. Which of the following is/are true about constructors in Java?

i) Constructor name should be same as class name.

ii) If you don't define a constructor for a class, a default parameter-less constructor is automatically created by the compiler.

iii) The default constructor calls super() and initializes all instance variables to default value like 0, null.

iv) If we want to parent class constructor, it must be called in first line of constructor.

1. i
2. i, ii
3. i, ii, iii
4. **i, ii, iii, iv**

2) Which type of Statement can execute parameterized queries?

1. **PreparedStatement**
2. ParameterizedStatement
3. ParameterizedStatement and CallableStatement
4. All kinds of Statements (i.e. which implement a sub interface of Statement)

3)Following is an example of prepared statement interface that?

PreparedStatement stmt=con.prepareStatement("select \* from emp");

ResultSet rs=stmt.executeQuery();

while(rs.next()){

System.out.println(rs.getInt(1)+" "+rs.getString(2));

}

1. deletes the record
2. **retrieve the record**
3. updates the record
4. inserts the record

4) What will happen when below code run?

public class JavaJDBC {

public static void main(String[] args) {

String dbUrl="jdbc:mysql://localhost/test";

String userName="root";

String passWord="";

Connection dbConn;

Statement myStmt = null;

dbConn = DriverManager.getConnection(dbUrl,userName,passWord);

myStmt=dbConn.createStatement();

myStmt.execute("DROP TABLE emp;");

}

}

1. **Checked Exception occur**
2. It will drop emp table.
3. Delete whole data from table but table will not be deleted
4. SQL statement is not correct

5) JDBC stands for:

1. **Java Database Connectivity**
2. Java Database Components
3. Java Database Control
4. None of the above is correct.

6) Which of the following encapsulates an SQL statement which is passed to the database to be parsed, compiled, planned and executed?

1. DriverManager
2. JDBC driver
3. Connection
4. **Statement**

7) Observe the following code carefully and determine the result:

import java.util.ArrayList;

import java.util.Iterator;

public class Test{

public static void main(String args[]) {

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

al.add(5);

al.add(10);

al.add(15);

al.add("H");

al.add(20);

al.add(25);

Iterator <Integer> i = al.iterator();

while(i.hasNext())

System.out.println(i.next());

}

}

1. **Compile-time error**
2. 5, 10, 15, H, 20, 25
3. 5, 10, 15, 20, 25
4. Run-time error

8) Identify output of following program.

public class MainClass{

public static void main(String args[]) {

T1.start();

}

}

class T1 extends Thread{

public void run() {

System.out.println("Thread executed.");

}

}

1. **compile-time error.**
2. run-time error.
3. Thread executed.
4. No error but blank output.

9) What will be the output of following code?

import java.io.\*;

public class Test {

public static void main(String[] args) {

String obj = "abc";

byte b[] = obj.getBytes();

ByteArrayInputStream obj1 = new ByteArrayInputStream(b);

for (int i = 0; i < 2; ++ i) {

int c;

while ((c = obj1.read()) != -1) {

if(i == 0) {

System.out.print((char)c + "-");

}

}

}

}

}

1. abc
2. a-b-c
3. **a-b-c-**
4. ABC

10) What is the output of this program?

public class string\_class {

public static void main(String args[]) {

String obj = "hello";

String obj1 = "world";

String obj2 = "hello";

System.out.println(obj.equals(obj1) + " " + obj.equals(obj2));

}

}

1. **false true**
2. false false
3. true true
4. true false

**2 Marks MCQs:**

11) What will be the output of the following program?

public class Test {

void m1() throws ArithmeticException {

throw new ArithmeticException("Calculation error");

}

void m2() throws ArithmeticException {

m1();

}

void m3() {

try {

m2();

}

catch (ArithmeticException e) {

System.out.println("ArithmeticException");

}

}

public static void main(String args[]) {

Test t = new Test();

t.m3();

System.out.println("Handled by JAVA");

}

}

1. Calculation error
2. ArithmeticException
3. Handled by JAVA
4. **ArithmeticException Handled by JAVA**

12) What will be the output of following code?

abstract class A{

abstract void firstMethod();

void secondMethod() {

System.out.println("SECOND");

firstMethod();

}

}

abstract class B extends A{

void firstMethod() {

System.out.println("FIRST");

thirdMethod();

}

abstract void thirdMethod();

}

class C extends B{

void thirdMethod() {

System.out.println("THIRD");

}

}

public class MainClass{

public static void main(String[] args) {

C c = new C();

c.firstMethod();

c.secondMethod();

c.thirdMethod();

}

}

1. Compilation error
2. **FIRST, THIRD, SECOND, FIRST, THIRD, THIRD**
3. THIRD, THIRD, SECOND, FIRST, FIRST, SECOND
4. SECOND, THIRD, THIRD, FIRST, FIRST, SECOND

13) What is the output of this program?

class recursion {

int func (int n) {

int result;

result = func (n - 1);

return result;

}

}

public class mcq {

public static void main(String args[]){

recursion obj = new recursion() ;

System.out.print(obj.func(12));

}

}

1. 0
2. 1
3. Compilation Error
4. **Runtime Error**

14) What is the output of the following program?

class Derived {

public void getDetails() {

System.out.println("Derived class");

}

}

public class Test extends Derived {

protected void getDetails() {

System.out.println("Test class");

}

public static void main(String[] args) {

Derived obj = new Test(); // line xyz

obj.getDetails();

}

}

1. Test class
2. Compilation error due to line xyz
3. Derived class
4. **Compilation error due to access modifier**

15) What is the output of this program?

import java.util.\*;

public class Maps

{

public static void main(String args[])

{

HashMap obj = new HashMap();

obj.put("A", new Integer(1));

obj.put("B", new Integer(2));

obj.put("C", new Integer(3));

System.out.println(obj.get("B"));

}

}

1. 1
2. **2**
3. 3
4. Null

**CODING: 5 Marks**

16) ***REDUCED STRING***

Steve has a string of lowercase characters in range ascii [‘a’..’z’]. He wants to reduce the string to its shortest length by doing a series of operations. In each operation he selects a pair of adjacent lowercase letters that match, and he deletes them. For instance, the string "aab" could be shortened to "b" in one operation.

Steve’s task is to delete as many characters as possible using this method and print the resulting string. If the final string is empty, print the original string as it is.

For e.g. "aabb" will produce and output of "aabb".

*Input Format*

First line of sample input represents t, test cases.

Next t lines are the input strings

*(Refer Sample Input below for explanation)*

*Output Format*

If the final string is empty, print the original string as it is; otherwise, print the final non-reducible string.

*Explanation*:

Steve performs the following sequence of operations to get the final string:

aaabccddd → abccddd → abddd → abd

*(Refer Sample Output below for explanation)*

*Constraints*

1<=len(s)<=100

*Sample Input 1*

2

aaabccddd

ssyy

*Sample Output 1*

abd

ssyy

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1  aabbc | 2  abbbbccdde  aabbccdd | 3  aabbccddeeffab  ssdduuttoo  sudoku | 4  aaccddeefghjkli  oppqqrty  mnbbjjjiiiiiyyyyytttt  preview | 5  aabbcceeffgghhjjkkiiuuyyttrree  statement  yahoo  hello  jammu |
| c | ae  aabbccdd | ab  ssdduuttoo  sudoku | fghjkli  orty  mnjiy  preview | aabbcceeffgghhjjkkiiuuyyttrree  statement  yah  heo  jau |

***Solution***:

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class file {

public static void main(String[] args) {

**Scanner s = new Scanner(System.in);**

**int t=s.nextInt();**

**for(int j=0;j<t;j++) {**

**String input = s.next();**

**StringBuilder str = new StringBuilder();**

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

**if(str.length() > 0 && str.charAt(str.length() - 1) == input.charAt(i)){**

**str.deleteCharAt(str.length() - 1);**

**}else{**

**str.append(input.charAt(i));**

**}**

**}**

**if(str.length() == 0){**

**System.out.println(input);**

**}else{**

**System.out.println(str);**

**}**

**}**

}

}

**17)** ***CUPCAKES***

Marc loves cupcakes, but he also likes to stay fit. Each cupcake has a calorie count, and Marc can walk a distance to expend those calories. If Marc has eaten j cupcakes so far, after eating a cupcake with c calories he must walk at least (2j) x c miles to maintain his weight.

For example, if he eats cupcakes with calorie counts in the following order: [5 10 7], the miles he will need to walk are ((20)x 5 + (21)x 10 + (22)x 7) = 5+20+28 =53. This is not the minimum, though, so we need to test other orders of consumption. In this case, our minimum miles is calculated as ((20) x 10 + (21) x 7 + (22) x 5) = 10+14+20=44.

Given the individual calorie counts for each of the cupcakes, determine the minimum number of miles Marc must walk to maintain his weight. Note that he can eat the cupcakes in any order.

*Input Format*

The first line contains an integer n, the number of cupcakes in calorie.

The second line contains n space-separated integers calorie[i].

*Output Format*

Print a long integer denoting the minimum number of miles Marc must walk to maintain his weight.

*Sample Input 1*

3

5 10 7

*Sample Output 1*

44

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5  2 4 5 1 3 | 2  91 199 | 4  11 22 33 44 | 7  12 10 21 9 19 34 56 | 8  112 425 267 1234 87 6788 76544 4532 |
| 57 | 381 | 286 | 1448 | 151768 |

***Solution***:

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class file {

**public static void main(String[] args) {**

**Scanner in = new Scanner(System.in);**

**int n = in.nextInt();**

**int[] calories = new int[n];**

**for(int calories\_i=0; calories\_i < n; calories\_i++){**

**calories[calories\_i] = in.nextInt();**

**}**

**Arrays.sort(calories);**

**long sum = 0;**

**for(int i = n-1; i>=0; i--) {**

**sum += (long)calories[i]\*Math.pow(2, (n-1-i));**

**}**

**System.out.println(sum);**

}

}

**CODING: 10 Marks**

**18)** ***GEMSTONES***

John has collected various rocks. Each rock has various minerals embedded in it. Each type of mineral is designated by a lowercase letter in the range ascii [a-z]. There may be multiple occurrences of a mineral in a rock. A mineral is called a gemstone if it occurs at least once in each of the rocks in John's collection.

Given a list of minerals embedded in each of John's rocks, display the number of types of gemstones he has in his collection.

For example, the array of mineral composition strings arr = [abc abc bc]. The minerals b and c appear in each composite, so there are 2 gemstones.

*Input Format*

First line of sample input represents t, test cases.

Next line consists of an integer n, the size of string array.

Each of the next n lines contains a string arr[i] where each letter represents an occurrence of a mineral in the current rock.

*(Refer Sample Input below for explanation)*

*Output Format*

Print the number of types of gemstones in John's collection. If there are none, print 0.

*(Refer Sample Output below for explanation)*

*Sample Input 1*

2

3

abcdde

baccd

eeabg

2

yuuhj

ioler

*Sample Output 1*

2

0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2  5  abcde  abcde  abcde  abcd  abc  3  qwerty  qwerty  qwerty | 2  2  qwertyuiopasdfghjklzxcvbnm  qwertyuiopasdfghjklzxcvbnm  4  mnbvcxzlkjhgfdsapoiuytrewq  mnbvcxzlkjhgfdsapoiuytrewq  mnbvcxzlkjhgfdsapoiuytrewq  mnbvcxzlkjhgfdsapoiuytrewq | 1  10  qwerty  qwer  qwer  qwerty  qwertyui  qwerty  qwerty  qwert  qwerty  qwert | 4  2  abc  def  2  yuio  plkjhj  3  sdsdsd  dsdsds  sssddd  4  ytrewq  hgfdsa  zxcvbnm  poiu | 2  20  qwertyuiopasdfg  qwertyuiopasdfg  qwertyuiopasdfg  qwertyuiopasdfg  qwertyuiopasdfg  qwertyuiopasdfg  qwertyuiopasdfg  qwertyuiopasdfg  qwertyuiopasdfg  qwertyuiopasdfg  qwertyuiopasdfg  qwertyuiopasdfg  qwertyuiopasdfg  qwertyuiopasdfg  qwertyuiopasdfg  qwertyuiopasdfg  qwertyuiopasdfg  qwertyuiopasdfg  qwertyuiopasdfg  qwertyuiopasdfg  10  hjklzxcvbnm  hjklzxcvbnm  hjklzxcvbnm  hjklzxcvbnm  hjklzxcvbnm  mnbvcxzlkjh  mnbvcxzlkjh  mnbvcxzlkjh  mnbvcxzlkjh  mnbvcxzlkjh |
| 3  6 | 26  26 | 4 | 0  0  2  0 | 15  11 |

Solution:

import java.io.\*;

import java.util.\*;

public class file {

static void countGemstones(int n,String array[]) {

**int[] arr = new int[26];**

**char a;**

**int b=0;**

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

**int[] arra = new int[26];**

**for(int j=0;j<array[i].length();j++) {**

**a = array[i].charAt(j);**

**b = (int)a-97;**

**arra[b]++;**

**if(arra[b]==1) {**

**arr[b]++;**

**}**

**}**

**}**

**int ans = 0;**

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

**if(arr[i]==n) {**

**ans++;**

**}**

**}**

**System.out.println(ans);**

}

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

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int t = Integer.parseInt(br.readLine());

for(int k=0;k<t;k++){

int n = Integer.parseInt(br.readLine());

String[] array = new String[n];

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

array[i] = br.readLine();

}

countGemstones(n,array);

}

}

}