**All MCQ’s manatory**

**Attempt as many programs as possible**

**Make sure you cover Maximum Unit test cases**

**1. Given:**

2. public class Bunnies {

3. static int count = 0;

4. Bunnies() {

5. while(count < 10) new Bunnies(++count);

6. }

7. Bunnies(int x) { super(); }

8. public static void main(String[] args) {

9. new Bunnies();

10. new Bunnies(count);

11. System.out.println(count++);

12. }

13. }

What is the result?

B. 10

**2. Given:**

2. public class Jail {

3. private int x = 4;

4. public static void main(String[] args) {

5. protected int x = 6;

6. new Jail().new Cell().slam();

7. }

8. class Cell {

9. void slam() { System.out.println("throw away key " + x); }

10. }

11. }

Which are true? (Choose all that apply.)

D. Compilation fails due to an error on line 5.

**3. Given:**

2. public class Fabric extends Thread {

3. public static void main(String[] args) {

4. Thread t = new Thread(new Fabric());

5. Thread t2 = new Thread(new Fabric());

6. t.start();

7. t2.start();

8. }

9. public static void run() {

10. for(int i = 0; i < 2; i++)

11. System.out.print(Thread.currentThread().getName() + " ");

12. }

13. }

Which are true? (Choose all that apply.)

A. Compilation fails.

**4. Given:**

2. class Feline { }

3. public class BarnCat2 extends Feline {

4. public static void main(String[] args) {

5. Feline ff = new Feline();

6. BarnCat2 b = new BarnCat2();

7. // insert code here

8. }

9. }

Which, inserted independently at line 7, compile? (Choose all that apply.)

C. if(b instanceof Feline) System.out.print("3 ");

**5. Given:**

2. public class Choosy {

3. public static void main(String[] args) {

4. String result = "";

5. int x = 7, y = 8;

6. if(x == 3) { result += "1"; }

7. else if (x > 9) { result += "2"; }

8. else if (y < 9) { result += "3"; }

9. else if (x == 7) { result += "4"; }

10. else { result += "5"; }

11. System.out.println(result);

12. }

13. }

What is the result? (Choose all that apply.)

A. 3

**6. Given:**

1. public class Twine {

2. public static void main(String[] args) {

3. String s = "";

4. StringBuffer sb1 = new StringBuffer("hi");

5. StringBuffer sb2 = new StringBuffer("hi");

6. StringBuffer sb3 = new StringBuffer(sb2);

7. StringBuffer sb4 = sb3;

8. if(sb1.equals(sb2)) s += "1 ";

9. if(sb2.equals(sb3)) s += "2 ";

10. if(sb3.equals(sb4)) s += "3 ";

11. String s2 = "hi";

12. String s3 = "hi";

13. String s4 = s3;

14. if(s2.equals(s3)) s += "4 ";

15. if(s3.equals(s4)) s += "5 ";

16. System.out.println(s);

17. }

18. }

What is the result?

E. 3 4 5

.

**7. Which are true? (Choose all that apply.)**

D. All classes of Exception extend Throwable.

E. All Throwables must be handled or declared.

G. RuntimeExceptions need never be handled or declared.

**8. Given:**

2. import java.util.\*;

3. public class Birthdays {

4. public static void main(String[] args) {

5. Map<Friends, String> hm = new HashMap<Friends, String>();

6. hm.put(new Friends("Charis"), "Summer 2009");

7. hm.put(new Friends("Draumur"), "Spring 2002");

8. Friends f = new Friends(args[0]);

9. System.out.println(hm.get(f));

10. }

11. }

12. class Friends {

13. String name;

14. Friends(String n) { name = n; }

15. }

And the command line invocation:

**# java Birthdays Draumur**

What is the result?

F. An exception is thrown at runtime.

**9. Given:**

2. import java.util.\*;

3. class Cereal { }

4. public class Flakes extends Cereal {

5. public static void main(String[] args) {

6. List<Flakes> c0 = new List<Flakes>();

7. List<Cereal> c1 = new ArrayList<Cereal>();

8. List<Cereal> c2 = new ArrayList<Flakes>();

9. List<Flakes> c3 = new ArrayList<Cereal>();

10. List<Object> c4 = new ArrayList<Flakes>();

11. ArrayList<Cereal> c5 = new ArrayList<Flakes>();

12. }

13. }

Which are true? (Choose all that apply.)

B. Compilation fails due to an error on line 6.

D. Compilation fails due to an error on line 8.

E. Compilation fails due to an error on line 9.

F. Compilation fails due to an error on line 10.

G. Compilation fails due to an error on line 11.

**10. Given:**

3. public class RediMix extends Concrete {

4. RediMix() { System.out.println("r "); }

5. public static void main(String[] args) {

6. new RediMix();

7. }

8. }

9. class Concrete extends Sand {

10. Concrete() { System.out.print("c "); }

11. private Concrete(String s) { }

12. }

13. abstract class Sand {

14. Sand() { System.out.print("s "); }

15. }

What is the result?

D. s c r

**11. Which statement(s) are true? (Choose all that apply.)**

B. Coupling is the OO principle most closely associated with making sure classes know about

other classes only through their APIs.

**12. Given:**

2. class Mosey implements Runnable {

3. public void run() {

4. for(int i = 0; i < 1000; i++) {

5. System.out.print(Thread.currentThread().getId() + "-" + i + " ");

6. } } }

7. public class Stroll {

8. public static void main(String[] args) throws Exception {

9. Thread t1 = new Thread(new Mosey());

10. // insert code here

11. }

12. }

Which of the following code fragments, inserted independently at line 10, will probably run

most (or all) of the main thread’s run() method invocation before running most of the t1

thread’s run() method invocation? (Choose all that apply.)

A. t1.setPriority(1);

new Mosey().run();

t1.start();

**13. Given:**

37. boolean b = false;

38. int i = 7;

39. double d = 1.23;

40. float f = 4.56f;

41.

42. // insert code here

Which line(s) of code, inserted independently at line 42, will compile and run without

exception? (Choose all that apply.)

A. System.out.printf(" %b", b);

D. System.out.format(" %d", i);

E. System.out.format(" %f", f);

**14. Given:**

1. import java.util.\*;

2. public class MyPancake implements Pancake {

3. public static void main(String[] args) {

4. List<String> x = new ArrayList<String>();

5. x.add("3"); x.add("7"); x.add("5");

6. List<String> y = new MyPancake().doStuff(x);

7. y.add("1");

8. System.out.println(x);

9. }

10. List<String> doStuff(List<String> z) {

11. z.add("9");

12. return z;

13. }

14. }

15. interface Pancake {

16. List<String> doStuff(List<String> s);

17. }

What is the most likely result?

D. Compilation fails.

**15. Given:**

3. import java.util.\*;

4. public class VLA2 implements Comparator<VLA2> {

5. int dishSize;

6. public static void main(String[] args) {

7. VLA2[] va = {new VLA2(40), new VLA2(200), new VLA2(60)};

8.

9. Arrays.sort(va, va[0]);

10. int index = Arrays.binarySearch(va, new VLA2(40), va[0]);

11. System.out.print(index + " ");

12. index = Arrays.binarySearch(va, new VLA2(80), va[0]);

13. System.out.print(index);

14. }

15. public int compare(VLA2 a, VLA2 b) {

16. return b.dishSize - a.dishSize;

17. }

18. VLA2(int d) { dishSize = d; }

19. }

What is the result?

D. 2 -2

**16. Given a directory structure:**

- baseDir

- testDir

- subDir2

- Shackelton.txt

and given the following code:

12. String name = "testDir" + File.pathSeparator + "subDir2"

+ File.pathSeparator + "Shackelton.txt";

13. File f = new File(name);

14. System.out.println("exists " + f.exists());

Assuming the proper import statements and exception handling, which statements must be true

in order for the output to be "exists true"? (Choose three.)

C. The program must be invoked from the baseDir directory.

**17. Given:**

1. import java.io.\*;

2. import java.util.\*;

3. import static java.lang.Short.\*;

4. import static java.lang.Long.\*;

5. public class MathBoy {

6. public static void main(String[] args) {

7. long x = 123456789;

8. short y = 22766; // maximum value of a short is 32767

9. System.out.printf("%1$+10d %2$010d ", x, MAX\_VALUE - y);

10. System.out.println(new Date());

11. }

12. }

Which are true? (Choose all that apply.)

A. Compilation fails.

**18. Given:**

1. public class WeatherTest {

2. static Weather w;

3. public static void main(String[] args) {

4. System.out.print(w.RAINY.count + " " + w.Sunny.count + " ");

5. }

6. }

7. enum Weather {

8. RAINY, Sunny;

9. int count = 0;

10. Weather() {

11. System.out.print("c ");

12. count++;

13. }

14. }

What is the result?

C. c c 1 1

**19. Given:**

2. import java.text.\*;

3. public class Gazillion {

4. public static void main(String[] args) throws Exception {

5. String s = "123.456xyz";

6. NumberFormat nf = NumberFormat.getInstance();

7. System.out.println(nf.parse(s));

8. nf.setMaximumFractionDigits(2);

9. System.out.println(nf.format(s));

10. }

11. }

Which are true? (Choose all that apply.)

C. The output will contain "123.456"

E. An exception will be thrown at runtime.

**20. Given that the current directory is bigApp, and the following directory structure:**

bigApp

|-- classes

|-- com

|-- wickedlysmart

|-- BigAppMain.class

And the code:

package com.wickedlysmart;

public class BigAppMain {

public static void main(String[] args) {

System.out.println("big app");

}

}

Which will invoke BigAppMain? (Choose all that apply.)

A. java classes/com.wickedlysmart.BigAppMain

G. java -cp classes/com/wickedlysmart com.wickedlysmart.BigAppMain

**21. Given:**

2. class Game {

3. static String s = "-";

4. String s2 = "s2";

5. Game(String arg) { s += arg; }

6. }

7. public class Go extends Game {

8. Go() { super(s2); }

9. { s += "i "; }

10. public static void main(String[] args) {

11. new Go();

12. System.out.println(s);

13. }

14. static { s += "sb "; }

15. }

What is the result?

E. Compilation fails.

**22. Given:**

2. public class Salmon extends Thread {

3. public static long id;

4. public void run() {

5. for(int i = 0; i < 4; i++) {

6. // insert code here

7. new Thread(new Salmon()).start();

8. throw new Error();

9. }

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

11. } }

12. public static void main(String[] args) {

13. Thread t1 = new Salmon();

14. id = t1.getId();

15. t1.start();

16. } }

And the two code fragments:

I. if(i == 2 && id == Thread.currentThread().getId()) {

II. if(i == 2) {

When inserting either fragment, independently at line 6, which are true? (Choose all that apply.)

Ans :None of them is true when considering both fragments

**23. Given:**

2. public class Internet {

3. private int y = 8;

4. public static void main(String[] args) {

5. new Internet().go();

6. }

7. void go() {

8. int x = 7;

9. TCPIP ip = new TCPIP();

10. class TCPIP {

11. void doit() { System.out.println(y + x); }

12. }

13. ip.doit();

14. }

15. }

What is the result? (Choose all that apply.)

D. Compilation fails due to an error on line 9.

**24. Given:**

4.

public static void main(String[] args) {

5. try {

6. if(args.length == 0) throw new Exception();

7. }

8. catch (Exception e) {

9. System.out.print("done ");

10. doStuff(); // assume this method compiles

11. }

12. finally {

13. System.out.println("finally ");

14. }

15. }

Which are possible outputs? (Choose all that apply.)

C. "done finally "

**25. Given:**

3. class A { }

4. class B extends A { }

5. class C extends B { }

6. public class Carpet<V extends B> {

7. public <X extends V> Carpet<? extends V> method(Carpet<? super X> e) {

8. // insert code here

9. } }

Which, inserted independently at line 8, will compile? (Choose all that apply.)

A. return new Carpet<X>();

B. return new Carpet<V>();

1. Validating IP addresses

You are a Network Admin for your company and your day-to-day activity includes interacting with IP addresses and network subnets. Today, you will focus only on the IPv4 protocol. You have a situation for which you need to determine if a given IP address belongs to a subnet. To do so, you must compare the candidate IP address with the range of the subnet. Consider an example subnet that begins at 10.0.0.1 and ends at 11.199.88.254. For this subnet, 10.43.59.96 is within range, while 111.19.12.154 is outside of the range. Note that your input may also contain invalid IP addresses such as 12500.58.18. Any IP address that is incorrectly formatted, is less than 0.0.0.0, or is greater than 255.255.255.255 is an invalid IP address.

In summary, your task is to determine whether an IP address is within range, out side of range, or invalid.

Input definition

Each line of the input will contain 3 IP Addresses, separated by spaces:

• The first IP address represents the first valid IP address of the subnet.

• The second IP address represents the   
last valid IP address of the subnet.

• The 3rd IP address is the address to validate.

Output definition

For each line of input, output the result in one of the following 3 ways (case sensitive):

• InRange

• InValid

• OutRange

Example input

10.0.0.1 11.199.88.254 1000.43.59.96

10.0.0.1 11.199.88.254 10.43.59.96

10.0.0.1 11.199.88.254 111.19.12.154

Example output

InValid

InRange

OutRange

2.

Unicode email addresses

With the expansion of the Internet across the world, the demand is growing for everyone to have an email address. Until everyone is ready to use international characters, you are now in charge of writing a converter that will take UTF-8 characters and bring them back to something DNS servers will be able to handle.

To do so, you must take an email string and break it into 3 pieces: first@second.third. Process each piece as follows:

1. Leave all characters that have ASCII codes between 33 and 126 as is, except convert uppercase letters to lowercase letters

2. Represent all other characters (referred to hereinafter as special characters) using the format +position?UTF-8\_hexadecimal\_representation

◦ The position is the 0-indexed position of the represented character

◦ The UTF-8 hexadecimal representation must not contain spaces and must be finalized

If at least one special character was present in the string, append a ~ and each special character at the end of the address.

For example, the piece Pøpø would be processed to pp+1?C3B8+3?C3B8~øø

Clarifications

• There shouldn’t be any spaces.

• You will receive only characters contained within the UTF-8 standard

Input definition

Each line will be an email address with Unicode characters.

Output definition

<new email address composition>~<characters removed in order>

Example input

WoRlD\_ReAdY@Microsoft.COM

Café@Chaud.tôt

Trinitŷ@߷.com

ա@մ.կ

😎@🚀.com

ep@ез.кom

Soup@🍚.com

🆍🆌@SAPA.com

ಠಠ@Look.com

ᱪ@Alien.invasion

Vikingoftheᛗ@North.ᛣ

ဩ@ဦ.com

Ⴭaჯ@Greorgian.letTer

Example output

world\_ready@microsoft.com

caf+3?C3A9@chaud.tt+1?C3B4~éô

trinit+6?C5B7@+0?DFB7.com~ŷ߷

+0?D5A1@+0?D5B4.+0?D5AF~ամկ

+0?EFBFBD+1?EFBFBD@+1?EFBFBD+2?EFBFBD.com~😎🚀

ep@+0?D0B5+1?D0B7.om+0?D0BA~езк

soup@+0?EFBFBD+1?EFBFBD.com~🍚

+0?EFBFBD+1?EFBFBD+2?EFBFBD+3?EFBFBD@sapa.com~🆍🆌

+0?E0B2A0+1?E0B2A0@look.com~ಠಠ

+0?E1B1AA@alien.invasion~ᱪ

vikingofthe+14?E19B97@north.+0?E19BA3~ᛗᛣ

+0?E180A9@+0?E180A6.com~ဩဦ

a+0?E1838D+2?E183AF@greorgian.letter~Ⴭჯ

3.

List out all the palindromes between 100 to 1000

4.

Given an array of names of candidates in an election. A candidate name in array represents a vote casted to the candidate. Print the name of candidate that received Max votes. If there is tie, print lexicographically smaller name.

Input:

The name of the candidates separated by a space. Each name represents one vote casted to that candidate.

Output:

Print the name of the candidate with the maximum votes, and also print the votes casted for the candidate. The name and votes are separated by a space.

Constraints:

1<=N<=1000

Example:

Input:

2

13

john johnny jackie johnny john jackie jamie jamie john johnny jamie johnny john

3

andy blake clark

Output:

john 4

andy 1

Explanation:

For testcase1: john has 4 votes casted for him, but so does johny. john is lexicographically smaller, so we print john and the votes he received.

For testcase2: We have 3 votes. All the candidates get 1 votes each. We print andy as it is lexicographically smaller.

5.

Given an array of integers. The task is to find a peak element in it. An array element is peak if it is not smaller than its neighbors. For corner elements, we need to consider only one neighbor. For example, for input array {5, 10, 20, 15}, 20 is the only peak element. For input array {10, 20, 15, 2, 23, 90, 67}, there are two peak elements: 20 and 90.

Note: There may be multiple peak element possible, in that case you may return any valid index.

Input:

The first line of input contains an integer T denoting the no of test cases. Then T test cases follow. Each test case contains an integer N. Then in the next line are N space separated values of the array.

Output:

For each test case output will be 1 if the index returned by the function is an peak index.

Constraints:

1<=N<=100

1<=A[]<=1000

6.

Sort an ArrayList containing Integers and remove all the duplicates.

Example:

Input

[20, 12, 15, 14, 8, 10, 11, 25, 12, 14, 10]

Output

[8, 10, 11, 12, 14, 15, 20, 25]