**SET A CORE JAVA (CSL 5303) RETEST ST1**

1. Which of these constructors is used to create an empty String object?

|  |  |  |  |
| --- | --- | --- | --- |
| (i) | **String()** | (ii) | String(void) |
| (iii) | String(null) | (iv) | String(0) |

1. The parent class of all exceptions in Java is \_\_\_\_\_\_\_\_\_\_\_\_.

|  |  |  |  |
| --- | --- | --- | --- |
| (i) | Throw | (ii) | Throwable |
| (iii) | Error | (iv) | **Exception** |

1. class className{

public static void main(String args[])

{

byte a = 126;

System.out.println(a);

a++;

System.out.println(a);

a++;

System.out.println(a);

a++;

System.out.println(a);

}

|  |  |  |  |
| --- | --- | --- | --- |
| (i) | 126 127 128 129 | (ii) | **126 127 -128 -127** |
| (iii) | -128 -127 -126 -125 | (iv) | 126 127 -127 -126 |

1. Which class is instantiable? Class A or Class B?

abstract class A{

}

class B extends A{

}

|  |  |  |  |
| --- | --- | --- | --- |
| (i) | Class A | (ii) | Neither class A nor class B |
| (iii) | **Class B** | **(iv)** | Both class A and B |

1. Which keyword is used to access the features of a package?

|  |  |  |  |
| --- | --- | --- | --- |
| (i) | **import** | (ii) | include |
| (iii) | export | (iv) | All of the above |

1. Give your answer for following code.

class Main

{

public static void main(String args[])

{

String str1="Retest";

String str2="Retest";

System.out.println(str1.equals(str2));

System.out.println(str1==str2);

}

|  |  |  |  |
| --- | --- | --- | --- |
| (a) | true  false | (b) | **true**  **true** |
| (c) | false  false | (d) | false  true |

1. Observe output of following:

public class Test{

public static void main(String args[]){

**int** x = 5;{

**int** x = 10;

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

}

}

}

|  |  |  |  |
| --- | --- | --- | --- |
| (a) | 5 | (b) | 10 |
| (c) | **Compiler Error** | (d) | Blank output |

1. Observe output of following code:

public class TestLoop2

{

public static void main(String... args)

{

int count = 10;

while( count++ < 11 )

System.out.println( count );

}

}

|  |  |  |  |
| --- | --- | --- | --- |
| (a) | **11** | (b) | 10 |
| (c) | Error in while loop | (d) | No output |

1. Which of these keywords is used to by the calling function to guard against the exception that is thrown by called function?

|  |  |  |  |
| --- | --- | --- | --- |
| (a) | try | (b) | **throws** |
| (c) | throw | (d) | catch |

1. What will be the output of following code?

**interface** I1{

**int** ***p***=10;

}

**public** **class** Mcq1 {

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

I1.***p***=12;

System.***out***.println(I1.***p***);

}

}

|  |  |  |  |
| --- | --- | --- | --- |
| (a) | **Compile time Error** | (b) | 10 |
| (c) | Run time Exception | (d) | 12 |

1. What will be the output of following code?

public class Mcq1 {

public static void throwit () {

System.out.print("throwit ");

throw new RuntimeException();

}

public static void main(String [] args) {

try {

System.out.print("hello ");

throwit();

}

catch (Exception re ){

System.out.print("caught ");

}

finally{

System.out.print("finally ");

}

System.out.println("after ");

}

}

|  |  |  |  |
| --- | --- | --- | --- |
| (i) | **hello throwit caught finally after** | (ii) | hello throwit caught |
| (iii) | hello throwit RuntimeException caught after | (iv) | Compilation fails |

1. What will be the output of following code?

abstract class X{

public X(){

System.out.println("ONE");

}

abstract void abstractMethod();

}

class Y extends X{

public Y(){

System.out.println("TWO");

}

void abstractMethod(){

System.out.println("THREE");

}

}

public class Mcq1{

public static void main(String[] args){

X x = new Y();

x.abstractMethod();

}

}

|  |  |  |  |
| --- | --- | --- | --- |
| (i) | THREE | (ii) | THREE  TWO  ONE |
| (iii) | THREE  ONE  TWO | (iv) | **ONE**  **TWO**  **THREE** |

1. What will be the output of following code:

public class If2

{

static boolean b1, b2;

public static void main(String [] args)

{

int x = 0;

if ( !b1 )

{

if ( !b2 )

{

b1 = true;

x++;

if ( 5 > 6 )

{

x++;

}

if ( !b1 )

x = x + 10;

else if ( b2 = true )

x = x + 100;

else if ( b1 | b2 )

x = x + 1000;

}

}

System.out.println(x);

}

}

|  |  |  |  |
| --- | --- | --- | --- |
| (i) | 0 | (ii) | 1 |
| (iii) | **101** | (iv) | 111 |

1. What would happen while executing following code?

interface Ap{

final static int a;

public void boon();

}

public class Mcq1 implements Ap {

public void boon() {}

public static void main(String ar[]) {

Mcq1 t = new Mcq1();

}

}

|  |  |  |  |
| --- | --- | --- | --- |
| (i) | Run time error | (ii) | **Compile time error** |
| (iii) | Will compile but not execute | (iv) | Will compile and execute |

1. What would be the output of following code:

class Temp {

private Temp(int data){

System.out.print(" Constructor called ");

}

protected static Temp create(int data){

Temp obj = new Temp(data);

return obj;

}

public void myMethod() {

System.out.print (" Method called ");

}

}

public class Test {

public static void main(String[] args) {

Temp obj = Temp.create(20);

obj.myMethod();

}

}

|  |  |  |  |
| --- | --- | --- | --- |
| (i) | Compiler error | (ii) | **Constructor called Method called** |
| (iii) | Runtime error | (iv) | None of the above |

**Coding:**

1. ***Substrings***

For a given a string S, count all contiguous substrings starting and ending with same character. Also print all such substrings.

*Input Format:*

First Line of sample input denotes T no. of test cases.

Next T lines represent the input strings.

*Output Format:*

Output all contiguous substrings starting and ending with same character and display count of such substrings.

*Sample Input:*

2

abcab

aba

*Sample Output:*

a

abca

b

bcab

c

a

b

7

a

aba

b

a

4

Explanation: There are 15 substrings of "abcab"

{a, ab, abc, abca, abcab, b, bc, bca, bcab, c, ca, cab, a, ab, b}

Out of the above substrings, there are 7 substrings: a, abca, b, bcab, c, a and b.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1  ababab | 2  bab  bca | 1  abcdef | 3  qq  www  eeeee | 2  rtyur  opq |
| a  aba  ababa  b  bab  babab  a  aba  b  bab  a  b  12 | b  bab  a  b  4  b  c  a  3 | a  b  c  d  e  f  6 | q  qq  q  3  w  ww  www  w  ww  w  6  e  ee  eee  eeee  eeeee  e  ee  eee  eeee  e  ee  eee  e  ee  e  15 | r  rtyur  t  y  u  r  6  o  p  q  3 |

Solution:

**import** java.util.\*;

**public** **class** file {

**static** **int** countSubstringWithEqualEnds(String s) {

**int** result = 0;

**int** n = s.length();

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

**for** (**int** len = 1; len <= n-i; len++)

**if** (*checkEquality*(s.substring(i, i + len))) {

result++;

System.***out***.println(s.substring(i, i + len));

}

**return** result;

}

**static** **boolean** checkEquality(String s) {

**return** (s.charAt(0) == s.charAt(s.length() - 1));

}

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

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

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

String str = "";

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

str = sc.next();

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

}

}

}

1. ***Array Sum***

Shahid is a computer science student and his teacher gave him a simple question to solve within time but he is little busy in his life, so he asked you to solve this problem.

You have been given an array of positive integers A1,A2,...,An with length N and you have to print an array of same length(N) where the values in the new array are the sum of every number in the array, except the number at that index.

***Sample Input:***

2

4

1 2 3 4

3

5 6 7

***Sample Output:***

9 8 7 6

13 12 11

***Input Format:***

* The first line of input contains a single integer T denoting the number of test cases.
* Each test cases contain two lines. First line contains N, the length of the array and second line contains N space separated positive integers.

***Output Format:***

For each test case, output a single array of same length.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2  5  0 0 0 0 0  3  -1 -1 -1 | 4  5  -1 1 3 4 0  2  1 -1  5  11 12 13 14 15  4  0 -1 1 -2 | 3  5  20 10 -11 8 6  3  11 11 11  3  9 5 7 | 3  1  9  2  1 6  3  100 1000 10000 | 1  4  5 -6 -11 12 |
| 0 0 0 0 0  -2 -2 -2 | 8 6 4 3 7  -1 1  54 53 52 51 50  -2 -1 -3 0 | 13 23 44 25 27  22 22 22  12 16 14 | 0  6 1  11000 10100 1100 | -5 6 11 -12 |

***Solution***:

**import** java.util.\*;

**public** **class** ArraySum

{

**static** **void** arraySum(**int**[] ar)

{

**int** sum=0;

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

sum+=ar[i];

}

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

{

System.***out***.print((sum-ar[i])+" ");

}

}

**public** **static** **void** main (String[] args) **throws** java.lang.Exception

{

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

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

**while**(t-->0)

{

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

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

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

ar[i]=ag.nextInt();

}

*arraySum*(ar);

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

}

}

}

1. ***Building Stairs***

John is working for ATR labs whose office is situated at seventh floor in new DLF building. As he doesn’t have much time for workout. He decided to go to his office by stairs instead of lift. He can either continuously climb stairs or wait at particular floor if he will feel tired. He will not use lift to jump some floors. Also While going back to home he will follow the same pattern in reverse order, so he can feel less fatigue. John is facing problem in memorizing which pattern he followed. Help him to recall whether he followed the same pattern while going and coming back from office or not. If he followed same pattern, then write yes otherwise print no. Also in case if John use lift then print no.

*Input Format:*

* The first line of the input contains an integer T denoting the number of test cases.
* The first line of each test case contains an integer N, denoting the number of elements in the given array.
* The second line contains N space-separated integers A1, A2, ..., AN denoting the elements of array.

*Output Format:*

For each test case, output a line containing "yes" or "no" (without quotes) corresponding to the case.

*Constraints:*

1 ≤ T ≤ 100

7 ≤ N ≤ 100

1 ≤ Ai ≤ 10

*Sample Input:*

3

19

1 2 3 4 4 5 6 6 6 7 6 6 6 5 4 4 3 2 1

14

1 2 3 4 5 6 7 6 5 4 3 2 1 1

13

1 2 3 4 5 6 8 6 5 4 3 2 1

*Sample Output:*

yes

no

no

*Explanation:*

The first example satisfies all the conditions.

The second example has 1 element of value 1 at the beginning and 2 elements of value 1 at the end.

The third one has no elements with value 7 after elements with value 6.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3  19  1 2 3 4 4 5 6 6 7 8 7 6 6 5 4 4 3 2 1  14  1 2 3 4 5 6 7 6 5 4 3 2 1 1  13  1 2 3 4 5 6 8 6 5 4 3 2 1 | 3  19  1 2 3 4 4 5 6 6 7 8 7 6 5 5 4 4 3 2 1  14  1 2 3 4 5 6 7 6 5 5 3 2 1 1  13  1 2 3 4 5 6 8 6 5 4 3 2 1 | 3  19  1 2 3 4 4 5 6 6 6 7 6 6 6 5 4 4 3 2 1  14  1 2 3 4 5 6 7 7 6 5 4 3 2 1  13  1 2 3 4 5 6 8 6 5 4 3 2 1 | 3  11  1 1 1 1 1 1 1 1 1 1 1  15  1 2 3 4 5 6 7 8 7 6 5 4 3 2 1  13  1 2 3 4 5 6 7 6 5 4 3 2 1 | 3  19  1 2 3 4 4 5 6 6 6 7 6 6 6 5 4 4 3 2 1  14  1 2 3 4 5 6 7 7 6 5 4 3 2 1  13  1 2 3 4 5 6 7 6 5 4 3 2 1 |
| no  no  no | no  no  no | yes  yes  no | no  no  yes | yes  yes  yes |

**Solution**:

**import** java.util.\*;

**import** java.lang.\*;

**import** java.io.\*;

**public** **class** file

{

**static** **void** climbStairs(**int** elementsArray[])

{

**int** flag=1,numberOfElements=elementsArray.length;

**int** j=numberOfElements-1,element=elementsArray[0];

**if**(elementsArray[0]!=1||elementsArray[numberOfElements/2]!=7 )

flag=0;

**else**

{

**for**(**int** k=0;k<=j&&flag==1;k++,j--)

{

**if**(elementsArray[k]!=elementsArray[j])

flag=0;

**if**(elementsArray[k]==element+1)

element++;

**else** **if**(element==elementsArray[k])

{}

**else**

flag=0;

}

}

**if**(flag==1)

System.***out***.println("yes");

**else**

System.***out***.println("no");

}

**public** **static** **void** main (String[] args) **throws** java.lang.Exception

{

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

**int** testCases=sc.nextInt();

**for**(**int** i=1;i<=testCases;i++){

**int** numberOfElements=sc.nextInt();

**int** elementsArray[]=**new** **int**[numberOfElements];

**for**(**int** j=0;j<numberOfElements;j++)

elementsArray[j]=sc.nextInt();

*climbStairs*(elementsArray);

}

}

}