## CSE111 Lab 10 – Exceptions, Applets, JavaFX and two IDEs

## Class Work (Task 1 to 5)

## Task 1

Create an array of size 5. Try to store 100 to the index 21 of the array. It should give you a runtime error. Note the first line of the error that was given.

You do not have to handle/catch anything for this task, just note the errors.

**Task 2:** In your main method, try to divide 1 by zero. Note the first line of the error that was given.

You do not have to handle/catch anything for this task, just note the errors.

## Task 3

import java.util.Scanner;

class Task3{

public static void main(String[]args){

Scanner sc=new Scanner(System.in);

int x,n=sc.nextInt();

int a[]=new int[n];

a[5]=99;

x=n/0;

}

}

When we enter 3 (or any number less than 6), then “a[5]=99;” line will cause error. Giving 6 or greater numbers will not cause that problem. Regardless of the number, “n/0” will always cause error.

Modify the program above to handle exception ‘**ArithmeticException’** and ‘

## ArrayIndexOutOfBoundsException’ using try, catch and finally keywords. Inside finally, just print the message “THE END”.

## Task 4

Run your solution of Task3 and give 2.5 as input. It will cause error.

Now you see that it is hard to know every error that may occur in advance. Hence, modify your solution to Task 3 to account for all other unknown errors that can be caused from any line of the main method.

## Task 5

Change and complete the MyReader class so that you can create your own exception ‘**EitaIntegerNoiException’**. There will be a method **int readInteger( )**. This method will take a String form the user. Then it will check if the number was ‘int’ or ‘float’ by checking presence of the decimal point in the input. It will throw an exception when a floating-point number is entered. Otherwise, it will return the integer number.

## Learning to use files (if java complains, use try catch blocks to handle exceptions or throw those)

**Task 6**

* Create a file named **a.txt** in your C drive root
* Write three numbers ( 14 15 16) on three separate lines
* Modify your solution of Task 1 (See rightmost column) by
  + Adding import **java.io.\*;** at the top (needed for the **File** class)
  + Replacing **Scanner s = new Scanner (System.in);**

with **String amarFileNameAndLocation = "c:\\a.txt";**

**File amarFile = new File ( amarFileNameAndLocation );**

**Scanner s =new Scanner (amarFile) );**

**Or, in short**

with **Scanner s =new Scanner (new File("c:\\a.txt") );**

* + Replacing **each int x = s.nextInt()**

with **String ektaLine;**

**ektaLine = s.nextLine();**

**int x = Integer.parseInt(ektaLine);**

* Run your program. It should give 45 as output.

**Hint**: Following two lines prints 12 on the screen:

int x=Integer.parseInt(**"**5**"**);

int y=Integer.parseInt(**"**7**"**);

System.out.println(x+y);

**Further reading:**

http://192.168.0.84/bucc/javadoc/api/java/lang/Integer.html#parseInt(java.lang.String)

**Task 7) Kaprekar Number**

Source: http://mathworld.wolfram.com/KaprekarNumber.html

Consider an -digit number . Square it and add the right digits to the left or digits. If the resultant sum is , then is called a Kaprekar number. For example, 9 is a Kaprekar number since



|  |
| --- |
|  |

and 297 is a Kaprekar number since

|  |
| --- |
|  |

The first few are 1, 9, 45, 55, 99, 297, 703, ...

Determine if a given number is **Kaprekar Number**

**You can test your solution from http://online-judge.uva.es/p/v9/974.html**

**Task 8) Odious Number**

http://mathworld.wolfram.com/OdiousNumber.html

In odious number is a nonnegative number that has an odd number of 1s in its binary expansion. The first few odious numbers are therefore 1, 2, 4, 7, 8, 11, 13, 14, 16, 19, ... (Sloane's [A000069](http://www.research.att.com/%7Enjas/sequences/A000069)). Numbers that are not odious are said to be [evil numbers](http://mathworld.wolfram.com/EvilNumber.html).

Determine if a given number is Odious or Evil number.

**Task 9) Narcissistic Number**

http://mathworld.wolfram.com/NarcissisticNumber.html

Check if a number is Narcissistic Number or not.

An -[digit](http://mathworld.wolfram.com/Digit.html) number that is the [sum](http://mathworld.wolfram.com/Sum.html) of the th [powers](http://mathworld.wolfram.com/Power.html) of its [digits](http://mathworld.wolfram.com/Digit.html) is called an -narcissistic number. It is also sometimes known as an Armstrong number, perfect digital invariant (Madachy 1979), or plus perfect number. Hardy (1993) wrote, "There are just four numbers, after unity, which are the sums of the cubes of their digits:



, , , and . These are odd facts, very suitable for puzzle columns and likely to amuse amateurs, but there is nothing in them which appeals to the mathematician." Narcissistic numbers therefore generalize these "unappealing" numbers to other [powers](http://mathworld.wolfram.com/Power.html) (Madachy 1979, p. 164).



The smallest example of a narcissistic number other than the trivial 1-[digit](http://mathworld.wolfram.com/Digit.html) numbers is

|  |  |
| --- | --- |
|  | (1) |

The first few are given by 1, 2, 3, 4, 5, 6, 7, 8, 9, 153, 370, 371, 407, 1634, 8208, 9474, 54748, ... (Sloane's [A005188](http://www.research.att.com/%7Enjas/sequences/A005188)).

|  |  |
| --- | --- |
|  | base-10 -narcissistic numbers |
| 1 | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 |
| 3 | 153, 370, 371, 407 |
| 4 | 1634, 8208, 9474 |
| 5 | 54748, 92727, 93084 |
| 6 | 548834 |
| 7 | 1741725, 4210818, 9800817, 9926315 |
| 8 | 24678050, 24678051, 88593477 |

**Task 10) The 3n + 1 problem**

http://online-judge.uva.es/p/v1/100.html

## The Problem

Consider the following algorithm:

1. input *n*

2. print *n*

3. if *n* = 1 then STOP

4. if *n* is odd then



5. else



6. GOTO 2

Given the input 22, the following sequence of numbers will be printed 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

It is conjectured that the algorithm above will terminate (when a 1 is printed) for any integral input value. Despite the simplicity of the algorithm, it is unknown whether this conjecture is true. It has been verified, however, for all integers *n* such that 0 < *n* < 1,000,000 (and, in fact, for many more numbers than this.)

Given an input *n*, it is possible to determine the number of numbers printed (including the 1). For a given *n* this is called the cycle-length of *n*. In the example above, the cycle length of 22 is 16.

For any two numbers *i* and *j* you are to determine the maximum cycle length over all numbers between *i* and *j*.

## 

## The Input

The input will consist of a series of pairs of integers *i* and *j*, one pair of integers per line. All integers will be less than 1,000,000 and greater than 0.

You should process all pairs of integers and for each pair determine the maximum cycle length over all integers between and including *i* and *j*.

You can assume that no operation overflows a 32-bit integer.

## The Output

For each pair of input integers *i* and *j* you should output *i*, *j*, and the maximum cycle length for integers between and including *i* and *j*. These three numbers should be separated by at least one space with all three numbers on one line and with one line of output for each line of input. The integers *i* and *j* must appear in the output in the same order in which they appeared in the input and should be followed by the maximum cycle length (on the same line).

## Sample Input

1 10

100 200

201 210

900 1000

## Sample Output

1 10 20

100 200 125

201 210 89

900 1000 174

**Task 11) Your Name in ASCII Art**

Using one 1-dimensional array of Strings or one 2-dimensional **character** array to print one part of your name having at least 5 characters.

### ## ## ## ## ### ## #### ### ########

## ## ### ## ### ## ## ## ## ## ## ## ##

## ## #### ## #### ## ## ## ## ## ## ## ##

## ## ## ## ## ## ## ## ## ## ## ## ## ## ##

######### ## #### ## #### ######### ## ## ## ######### ##

## ## ## ### ## ### ## ## ## ## ## ## ## ##

## ## ## ## ## ## ## ## ###### #### ## ## ##

Feel free to choose your own method of storing data in the array but you have to print using loop(s). Printing manually by mentioning numeric value of index is NOT accepted.

Whatever output your program gives, if it is copied to MS word and viewed with “Courier New” font and a size of 10 points (pts), it should look similar to the above example.

**Hint:** Use “banner3” font with your name at http://www.network-science.de/ascii/

**Task 12**

Write an abstract class ‘**Instrument’** which will have abstract method ‘**play’**, ‘**adjust’** & concrete method ‘**compose’**.

Use the abstract class ‘Instrument’ to create class ‘**Guitar’**, ‘**Keyboard’** & ‘**Violin’**.

Create instance (object) of every classes invoking(calling) every method. The method will print any message with ‘Instrument name’ and ‘Purpose’,

* The method ‘play’ for ‘Violin’ class will print “In the playing method of Violin”

**Task 13**

Write a java application as follows:

Create a Student Interface ‘**StInterface’** with the methods ‘**setName’**, ‘**setID’**, ‘**getName’** **and ‘getID’**.

Create the class ‘**Student’** with ‘name’, ‘id’ and ‘address field’ implementing the ‘**StInterface’** to manipulate the Student information using the necessary methods.

Create an array of objects of **Student**. Then input the number of students to allocate student array dynamically and take Student information. Now print the student list alphabetically.

Register (Create an account / become a member) at USA Computing Olympiad (USACO), <http://ace.delos.com/usacogate>

**Problem 14)**

Submit USACO’s “test” problem which adds two numbers and prints the summation.

**Problem 15)**

Solve USACO’s first problem, “PROB Your Ride Is Here”

Register (Create an account / become a member) at http://uva.onlinejudge.org

and then solve following problems

**Problem 16)** [10055 - Hashmat the Brave Warrior](http://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=94&page=show_problem&problem=996)

**Problem 17)** [10071 - Back to High School Physics](http://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=94&page=show_problem&problem=1012)

**Problem 18)** [10300 - Ecological Premium](http://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=94&page=show_problem&problem=1241)

**Problem 19)** [458 - The Decoder](http://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=94&page=show_problem&problem=399)

**Problem 20)** [494 - Kindergarten Counting Game](http://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=94&page=show_problem&problem=435)

**HW 1**

Modify your **Task 6** so that if a.txt file contains any number of lines each with one number, your program should sum all numbers and give correct output. **Hint:** hasNextLine() method tells if the file has any line left. For more, read http://192.168.0.84/bucc/javadoc/api/java/util/Scanner.html

**Example:**

String line;

while(s.hasNextLine()){

line = s.nextLine();

System.out.println( line );

}

**HW 2**

Modify your **Task 6** or HW1 so that your program takes input from b.txt file. That file will contain three numbers (16 17 18) but on the same line. Output should be 51. **Hint:** use the method **next() and hasNext()** instead of **nextLine()** and **hasNextLine();**

**HW 3**

Modify your **Task 6** or HW1 so that your program takes input from b.txt file. That file will contain three numbers (16 17 18) but on the same line. Output should be 51. **Hint:** use the method **nextInt() and hasNextInt()** instead of **nextLine()** and **hasNextLine();**

**HW 4**

Write a program that asks the user for file name. Then prints the whole file on screen line by line. Hint: read each line and immediately print that line.

**HW5**

Given a file name, delete that file. **Hint:** read

http://192.168.0.84/bucc/javadoc/api/java/io/File.html#delete()

http://www.java2s.com/Code/Java/File-Input-Output/DeletefileusingJavaIOAPI.htm

**HW6 (Encryption)**

Given a line as keyboard input in small letters, print the next alphabet in sequence for each alphabet found in the input

Sample Input 1:

abcd

Sample output 1:

bcde

Sample Input 2:

the cowz

Sample output 2:

uif dpxa

**HW7 (Decryption)**

Given a line as keyboard input in small letters, do the opposite of HW5

Sample Input 1:

bcde

Sample output 1:

abcd

Sample Input 2:

uif dpx

Sample output 2:

the cow

**Optional/Bonus Task 8**

One of my GRE (Graduate Record Exam.) question was about “Too much emphasis is placed on role models. Instead of copying others, people should learn to think and act independently and thus make the choices that are best for them.”. I answered that role models are necessary but instead of acting like them, we have to learn what contributed to their success (thoughts, education, exploration) and what did not (e.g. life style)

I came to know about Dr. Regina Dugan from MSA probably during Spring 2009. She is the director of Defense Advanced Research Projects Agency (DARPA), US Military. To inspire yourself towards self learning, read on her from

* http://www.darpa.mil/directorbio.html
* http://www.duganventures.com/team.html
* http://topics.nytimes.com/topics/reference/timestopics/people/d/regina\_e\_dugan/index.html

**HW 9 (Optional Bonus: This may repair part of your marks lost in earlier labs)**

**Mutant Flatworld Explorers**

http://online-judge.uva.es/p/v1/118.html

**Hint:** It is similar to Task 1 and 2 but here you will have to maintain a two dimensional character array

## 

## Background

Robotics, robot motion planning, and machine learning are areas that cross the boundaries of many of the subdisciplines that comprise Computer Science: artificial intelligence, algorithms and complexity, electrical and mechanical engineering to name a few. In addition, robots as ``turtles'' (inspired by work by Papert, Abelson, and diSessa) and as ``beeper-pickers'' (inspired by work by Pattis) have been studied and used by students as an introduction to programming for many years.

This problem involves determining the position of a robot exploring a pre-Columbian flat world.

## The Problem

Given the dimensions of a rectangular grid and a sequence of robot positions and instructions, you are to write a program that determines for each sequence of robot positions and instructions the final position of the robot.

A robot position consists of a grid coordinate (a pair of integers: x-coordinate followed by y-coordinate) and an orientation (N,S,E,W for north, south, east, and west). A robot instruction is a string of the letters 'L', 'R', and 'F' which represent, respectively, the instructions:

* Left: the robot turns left 90 degrees and remains on the current grid point.
* Right: the robot turns right 90 degrees and remains on the current grid point.
* Forward: the robot moves forward one grid point in the direction of the current orientation and maintains the same orientation.

The direction North corresponds to the direction from grid point (*x*,*y*) to grid point (*x*,*y*+1).

Since the grid is rectangular and bounded, a robot that moves ``off'' an edge of the grid is lost forever. However, lost robots leave a robot ``scent'' that prohibits future robots from dropping off the world at the same grid point. The scent is left at the last grid position the robot occupied before disappearing over the edge. An instruction to move ``off'' the world from a grid point from which a robot has been previously lost is simply ignored by the current robot.

## Hint: For your convenience, you may mark cells having the scent with ‘X’ or any character you like to mean forbidden cells.

## The Input

The first line of input is the upper-right coordinates of the rectangular world, the lower-left coordinates are assumed to be 0,0.

The remaining input consists of a sequence of robot positions and instructions (two lines per robot). A position consists of two integers specifying the initial coordinates of the robot and an orientation (N,S,E,W), all separated by white space on one line. A robot instruction is a string of the letters 'L', 'R', and 'F' on one line.

Each robot is processed sequentially, i.e., finishes executing the robot instructions before the next robot begins execution.

Input is terminated by end-of-file.

You may assume that all initial robot positions are within the bounds of the specified grid. The maximum value for any coordinate is 50. All instruction strings will be less than 100 characters in length.

## The Output

For each robot position/instruction in the input, the output should indicate the final grid position and orientation of the robot. If a robot falls off the edge of the grid the word ``LOST'' should be printed after the position and orientation.

## Sample Input

5 3

1 1 E

RFRFRFRF

3 2 N

FRRFLLFFRRFLL

0 3 W

LLFFFLFLFL

## Sample Output

1 1 E

3 3 N LOST

2 3 S

## Bonus Task 10

Prepare an applet that prints hello world or your name. Your teacher will not help you with this task.

Take help from: http://www.realapplets.com/tutorial/

Some interesting applet examples: http://icpcres.ecs.baylor.edu/

## HW 11 Execute (Run) MSWord and Dr.Java from your java program

**Help:**

http://intekhabsadekin.wordpress.com/2009/03/05/java-runtimegetruntimeexec-will-and-will-not-work/

## HW 12 Using Eclipse, Take 2 numbers as input, print the sum. See notes.

## HW 13 Using NetBeans, Take 2 numbers as input, print the multiplication. See notes.

## HW 14 Print hello world or your name using JavaFX (this task will recover previously lost marks)

**Notes for HW 12 and 13:**

Show all screen shots including how to create project, write class, compile, run and test your program.

You can collect downloaded copy of Eclipse and NetBeans from <http://192.168.0.84/bucc/Software/Programming>

Open the folder called Editors – IDE.

Alternatively, you can download from the Internet. Search using “download eclipse”.