Note: You can define user defined exception classes to handle the exceptional situations, in appropriate cases.

1) Write an OOPs program to build a calculator for basic arithmetic operations. The calculator accepts only integer inputs. In certain cases, other characters may be entered as input. Implement the calculator such that if the input entered is not a digit[0-9], then it should continue with the appropriate exception handling. Also, handle the incorrect operation “division by zero” in your code.

Input format:
3 lines of input.
1st line is the operator.
2nd line 1st argument.
3rd line 2nd argument.

Output format:
Print result if operation is valid. Else, show the appropriate exception.

Sample Input and Output:

Input:
+
5
7

Output:
12

Input:
/
10
0

Output:
ArithmeticException: Division by zero handled. Enter a valid argument.

2) Write a method ***printLine()*** which takes a filename as input. The method should print all those lines in the file which contains only digits[0-9]. If a line contains any character other than the digits, it should print “OtherCharactersFound” and continue reading from the file.
Note: Create a text file and read that file as input. Also handle exceptions such as FileNotFoundException and NumberFormatException (for non-numeric characters).

Input format:
Filename

Output format:
depends on file content
Sample Input and Output:
Consider “file1.txt” having the following content:
964562365364564387
8986White432%#$
54552
Black

Input:
file1.txt

Output:
964562365364564387
OtherCharactersFound
54552
OtherCharactersFound

3) Write a Java code which takes an array as input and prints the sum of the array elements at specific indexes entered by the user. If indexes are valid, print the sum. Otherwise, handle the ArrayIndexOutOfBoundsException.

Input format:
1st line : number of elements in the array
2nd line: Elements in the array.
3rd line : Index values. (Take 0 based indexing)

Output format:
Sum of the array elements at the indexes entered.

Sample Input and Output:

Input:
5
4 5 9 11 100
0 2 4

Output:
113

Input:
5
4 5 9 11 100

0 2 5

Output:

ArrayIndexOutOfBoundsException. Enter valid index.

4) Over the network, messages are sent as data packets. Each data packet is a series of bytes (8bits=1 byte). Write a java program to decode a message from a data packet and handle the error if there are extra bits in the packet. The format of the data packet is as follows:

- The packet can be broken into 2 parts.
- The first byte represents the length of the data packet.
- Remaining bytes in the data packet represent the message to be decoded.
- Each message byte is to be XORed to the first byte, which results in a set of alphanumeric ASCII characters to get the readable message.

The goal of the program is to decode the input packet and print the actual message sent across if the number of bits is a multiple of 8. Otherwise, handle the situation and print “Incorrect Input”.

Input format :

One line which contains the series of bits.

Output format :

One line output which prints the decoded ASCII characters from the input bit series. Print “Incorrect Input” if the number of bits is not a multiple of 8.

.

Sample Input:

Input:

10101010111000101100111111000110110001101100010 1

Output:

Hello

Input:

1010101011100010111000101111

Output:

Incorrect Input

5) Consider a checkerboard of size 8x8. Each square in the board can be defined as i*j where i and j ranges from 1 to 8. You are given the initial position on the board along with some movements. Write a Java program to find the final position after the movements. If the final position goes beyond the chessboard, handle the exception and print the message “Overflow”. Otherwise, print the final position as [i,j].

Input Format :
First line specifies the initial position.(row_number column_number)
Second line defines ‘N’ (the number of movements).
Third line has the ‘N’ movements separated by space. (Movements are Left, Right, Up, Down)
Output Format :
One line of output which shows the final position if it exists, otherwise print “Overflow”.

Sample Input and Output:

Input:
5 3
4
Up 2
Right 3
Down 5
Left 1

Output :
[8,5]

Input :
3 1
1
Left 2

Output :
Overflow

6) Create a class “Queue” with two operations enqueue() and dequeue(). Use enqueue() to insert elements into a queue and dequeue() to delete elements from the queue. During execution, handle the underflow exception condition by printing “EmptyQueue”.

Input Format :
One line input:
● If the operation is enqueue() then add the given element to the queue list.
● If the operation is dequeue() then delete the element from the queue list.

Output Format :
One line output:
● If the operation is enqueue(), print “Success”.
● If the operation is dequeue(), print the element, if an exception occurs then prin“EmptyQueue”.

Sample Input and Output :

Note: E indicates enqueue and D indicates dequeue

Input:
E 10
E 34
E 60
D
D
D
D

Output:
Success
Success
Success
10
34
60
EmptyQueue

7) Write a java program to take a hexadecimal value from the user and convert it into the corresponding decimal value. If the Hexadecimal value is invalid, throw “InvalidHexdecimal Exception”.

Input Format:
A Hexadecimal number

Output Format:
The decimal value or “InvalidHexadecimal” message

Sample Input and Output :

Input:
DC24
Output:
56356

Input:
24G
Output:
InvalidHexadecimal