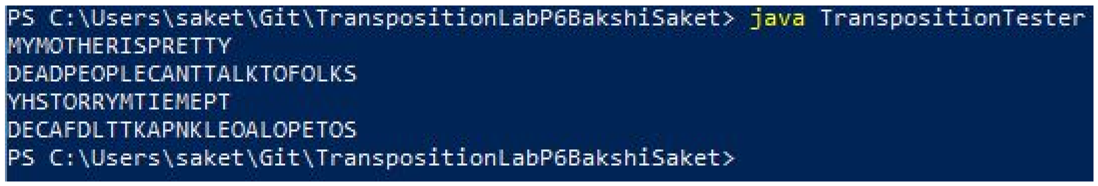
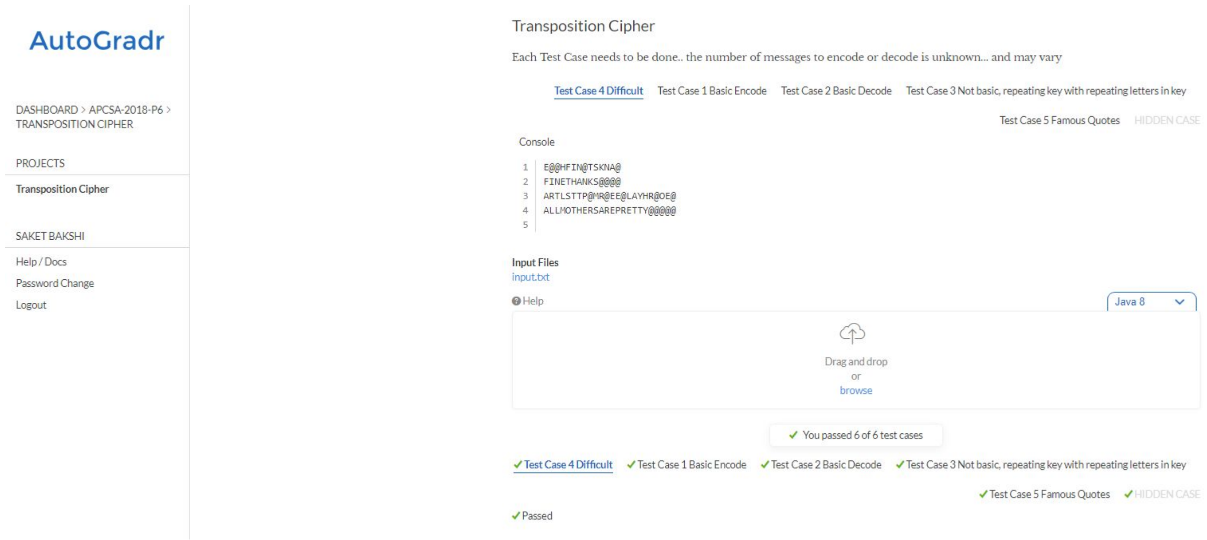


```
PS C:\Users\saket\Git\TranspositionLabP6BakshiSaket> java TranspositionTester
MYMOTHERISPRETTY
DEADPEOPLECANTTALKTOFOLKS
YHSTORRYMTIEMEPT
DECAFDLTTKAPNKLEOALOPETOS
PS C:\Users\saket\Git\TranspositionLabP6BakshiSaket>
```



Transposition Cipher

Each Test Case needs to be done.. the number of messages to encode or decode is unknown... and may vary

Test Case 4 Difficult   Test Case 1 Basic Encode   Test Case 2 Basic Decode   Test Case 3 Not basic, repeating key with repeating letters in key   Test Case 5 Famous Quotes   HIDDEN CASE

Console

```
E@@HFIN@TSKNA@
FINETHANKS@@@@
ARTLSTTP@MR@EE@LAYHR@OE@
ALLMOTHERSAREPRETTY@@@@@
```

Input Files
input.txt

✓ You passed 6 of 6 test cases

✓ Test Case 4 Difficult   ✓ Test Case 1 Basic Encode   ✓ Test Case 2 Basic Decode   ✓ Test Case 3 Not basic, repeating key with repeating letters in key   ✓ Test Case 5 Famous Quotes   ✓ HIDDEN CASE

✓ Passed

```
/* Saket Bakshi. AP Computer Science A. Transposition Lab. Due December 16, 2018.
This class takes input with instructions to encode or decode; a key; and a message. It encodes
or decodes according to the transposition cipher.
*/
public class TranspositionLabP6BakshiSaket
{
        private String encodeOrDecode; //tells to encode or decode
        private String key; //for the key
        private String inputMessage; //for whole message
        private String output; //output

        private String[][] tableArray; //array to hold message
                private int rows; //rows in array
                private int columns; //columns in array
        private int[] keyOrder; //array of the length of the key; shows alphabetical order of key

        /** This class takes input with instructions to encode or decode; a key; and a message. It
encodes or decodes according to the transposition cipher.
        @param encodeOrDecode the instructor to decode or encode
        @param key the key to the cipher
        @param inputMessage the message to encrypt or decrypt
        */
        public TranspositionLabP6BakshiSaket(String encodeOrDecode, String key, String
inputMessage)
```

```
	{
		this.encodeOrDecode = encodeOrDecode.toUpperCase();
		this.key = key.trim();
		this.inputMessage = inputMessage.replaceAll("[^a-zA-Z@]","");
			while(this.inputMessage.length() % this.key.length() != 0)
				this.inputMessage = this.inputMessage + "@";

		this.columns = this.key.length();
		this.rows = this.inputMessage.length() / this.key.length();
		this.tableArray = new String[this.rows][this.columns];
		this.keyOrder = new int[this.key.length()];
		this.output = "";
	}

	/** This method encrypts or decrypts the code. It finds the alphabetical order of the key,
adds "@" symbols as necessary to the input message, creates a suitable array, and either
encrypts or decrypts.
	*/
	public void runThroughInput()
	{
		//getting alphabetical order of the key
			char[] keyCharArray = this.key.toCharArray();
			int[] keyArray = new int[this.columns]; //array for putting int values of each
char in the key in their respective indexes
			for(int a = 0; a < this.columns; a++)
			{
				keyArray[a] = (int)keyCharArray[a];
			}
			for(int a = this.columns - 1; a >= 0; a--) //puts in alphabetical order of the
key
			{
				int maximumValue = -1000;
				int currentIndexOfKey = 0;
				for(int b = this.columns - 1; b >= 0 ; b--)
				{
					if(keyArray[b] > maximumValue) //finds the first letter in
alphabetical order
					{
						maximumValue = keyArray[b]; //sets letters as the
first
						currentIndexOfKey = b; //saves the letter's index
					}
				} //repeats to find true first alphabet
```

```
                        this.keyOrder[currentIndexOfKey] = a + 1; //sets alphabetical order
in the constructor array
                        keyArray[currentIndexOfKey] = -3000; //takes away letters as they
are placed alphabetically
                        maximumValue = -1000; //resets int object
                    }

                //encrypts or decrypts
                if(encodeOrDecode.equals("E")) //if encrypting
                {
                    int messageIndex = this.inputMessage.length();
                    for(int a = this.rows - 1; a >= 0; a--) //fills in tableArray with the message
                    {
                        for(int b = this.columns - 1; b >= 0; b--)
                        {
                            this.tableArray[a][b] =
this.inputMessage.substring(messageIndex - 1, messageIndex);
                            messageIndex--;
                        }
                    }
                    for(int a = 1; a <= this.columns; a++) //goes in alphabetical order of key
                    {
                        int keyIndex = 0;
                        while(this.keyOrder[keyIndex] != a) //shifts key array to match
alphabetical order
                            keyIndex++;
                        for(int b = 0; b < this.rows; b++) //goes down the row of the array
                        {
                            String temporaryCharacter = this.tableArray[b][keyIndex];
                            this.output = this.output + temporaryCharacter;
                        }
                    }
                }
                else if(encodeOrDecode.equals("D")) //if decrypting
                {
                    int messageIndex1 = 0;
                    for(int a = 1; a <= this.columns; a++) //goes in alphabetical order of key
                    {
                        int keyIndex = 0;
                        while(this.keyOrder[keyIndex] != a) //shifts key array to match
alphabetical order
                            keyIndex++;
                        for(int b = 0; b < this.rows; b++) //goes down the row of the array
                        {
                                String temporaryCharacter =
this.inputMessage.substring(messageIndex1, messageIndex1 + 1); //goes in order of message
                                messageIndex1++; //goes to next character in message
                                this.tableArray[b][keyIndex] = temporaryCharacter; //adds
character to the array
                        }
                }
                for(int a = this.rows - 1; a >= 0; a--)
                {
                        for(int b = this.columns - 1; b >= 0; b--) //goes through array
columns, then rows, adding characters in array to the output
                        {
                                String temporaryCharacter = this.tableArray[a][b];
                                this.output = temporaryCharacter + this.output; //takes
value of the character at point in array and adds to output
                        }
                }
        }
    }

    /** returns the output
    @return the output
    */
    public String getOutput()
    {
        return this.output;
    }
}
/* Saket Bakshi. AP Computer Science A. Transposition Lab. Due December 16, 2018.
This class takes tests the TranspositionLab class.
*/
import java.util.Scanner;
import java.io.File;
import java.io.FileNotFoundException;

public class TranspositionTester
{
    public static void main(String[] args) throws FileNotFoundException
    {
        File inFile = new File("input.txt"); //take in file
        Scanner scan = new Scanner(inFile); //create Scanner class object for file
```

```
		while(scan.hasNextLine())
		{
			String eOrD = scan.next(); //read in parts of Transposition Lab class to
construct an object
			String key = scan.next();
			String message = scan.nextLine();
			TranspositionLabP6BakshiSaket test = new
TranspositionLabP6BakshiSaket(eOrD, key, message); //construct the object
			test.runThroughInput(); //encrypt or decrypt
			System.out.println(test.getOutput()); //print the output
		}

		scan.close(); //closes the scanner
	}
}
```