JAVA PROGRAMMING
(340)

REGIONAL – 2014

TOTAL POINTS _____ ***(340 points)***

Judges/Graders: Please double check and verify all scores and answer keys!

:

Application/Execution • Application reads the names of the files from the command line	_____ 30 pts
• Application reads from the input text file	_____ 30 pts
• Applications correctly displays to the screen the original ciphertext/key pair and the corresponding plain text	_____ 40 pts
• Application correctly converts the ciphertext into plain text	_____ 60 pts
• Application reads the entire file (ciphers.txt)	_____ 30 pts
• Application correctly writes the ciphertext/key pair and corresponding plain text to the output text file (plain.txt)	_____ 40 pts
• Application does not display any extra results from reading past the file	_____ 20 pts
• Application does not write any extra results to the "plain.txt" file	_____ 20 pts
I/O error handling is done if files cannot be opened	_____ 10 pts
I/O error handling is done if no command arguments	_____ 10 pts
JavaDoc comments are used	_____ 10 pts
Code uses a class for the conversion	_____ 20 pts
Methods are commented	_____ 10 pts
Code copied to USB drive and program runs from USB	_____ 10 pts

Total Points: _____ 340 pts

Note to Grader: If you would like to run the java program, place the input text file in the root of the application where it can be located by the application.

Sample Solution code. Contestant code may vary. The class code is given first.

```
/**
 * @author contestant# <br />
 * This class will convert ciphertext into plain text
 *
 */
public class cipher {

        /**
         * This method takes the ciphertext character and the key character and converts it to plain text.
         * @param key a character of the key
         * @param code a character of ciphertext
         * @return <br />the plain text character
         */
        public char convert(char key, char code){

                        return sub(key,code);

        }


        private static char sub(char a, char b){  //calc a-b
     int ia=a, ib=b;
     if(ia<=ib) ia+=26;
     return (char) ('A'-1 +(ia-ib));
   }
}
import java.util.*;
import java.io.*;

public class CipherTest {

        //This is the main method that will read and write the files and
        //it will also call the cipher class to convert the cipher.

        public static void main(String[] args)throws IOException {
                String fin="";
                String fout="";
                cipher c= new cipher();
                if (args.length < 2){
                        System.out.println("Usage Error: Not enough Arguments.");
                        System.exit(0);
                }

                else {
```


```
                fin=args[0];
                fout=args[1];
        }

        try {
                BufferedReader br = new BufferedReader(new FileReader(fin));
                FileWriter fp = new FileWriter(fout);
                char convert;
                while(true){
                        String code = br.readLine(), key = br.readLine();
                        if (code!=null) {
                                System.out.print(code + "/" + key + " = ");
                                fp.write(code + "/" + key + " = ");
                        }
                        if(key==null) break;

                        while(key.length()<code.length()) key = key + key; //extra long
                        for(int i=0; i<code.length(); i++) {
                                convert=c.convert(code.charAt(i),key.charAt(i));
                                System.out.print(convert);
                                fp.write(convert);
                        }

                        System.out.println("");
                        fp.write("\n");


                }
                fp.close();
                br.close();
        }
        catch (FileNotFoundException e){
                System.out.println(fin +" file not found");
        }
    }
}
```

Note to Grader: When running the application, use the *grader_cipher.txt* file provided. The following is the correct output that should be displayed on the screen and out to the file:

The following is the correct output:

KFFOP/CAT = HELLO
JXK/FIDO = DOG
JPWIWTHBRPPDPNFHBRPP/ABCDEFGHI = INTERNATIONALIZATION
USPVCMF/A = TROUBLE
USPVCMF/AAAAAAA = TROUBLE
B/BCDEFGHIJ = Z