

**THE BISHOP'S SCHOOL – CAMP**

**COMPUTER PROJECT**

**REPORT**

**Implementation of**

**Cryptography in Java**

**Name: Omkar S. Nath**

**Class: 10 – D**

**Roll Number: 35**

**Computer Code: 230438**

**THE BISHOP'S SCHOOL, CAMP, PUNE.**



# **CERTIFICATE**

Roll Number: **35**

This is to certify that **Mr. Omkar S. Nath** of class **10-D** has satisfactorily completed the Computer Project for fulfillment of Indian Certificate of Secondary Education (ICSE) as prescribed by Indian School Certificate Examination for the academic year **2014-2015**.

\_\_\_\_\_  
Project Guide

\_\_\_\_\_  
External Examiner

\_\_\_\_\_  
Headmaster

\_\_\_\_\_  
Principal

# ACKNOWLEDGEMENT

I would like to thank my computer teachers, Mr. Shehrevan Davierwala and Mrs. Amruta Sarode, for their teaching and guidance throughout this project.

I would like to thank my headmaster, Mr. J. Edwin, and my principal, Mr. F. R. Freese, for managing the school and providing an enriching school environment.

I would like to thank my parents for providing me with the necessary motivation and support.

# INDEX

Serial Number	Topic	Page Number
1.	Certificate	2
2.	Acknowledgement	3
3.	Index	4
4.	Introduction	5 – 7
5.	About The Program	8 – 9
6.	Algorithm Of Ciphers	10 – 34
7.	System Requirement	35
8.	Source Code	36 – 222
9.	Output	223 – 265
10.	Screenshots	266 – 275
11.	Conclusion	276
12.	Future Enhancement	277
13.	Bibliography	278
14.	CD Containing Program	279

# INTRODUCTION

## **JAVA:**

Java is a computer programming language that is class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers ‘write once, run anywhere’ (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java applications are typically compiled to bytecode (class file) that can run on any Java virtual machine (JVM) regardless of computer architecture.

Java is, as of 2014, is one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers.

Java was originally developed by James Gosling at Sun Microsystems (which has since merged into Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

## **BLUEJ:**

BlueJ is an integrated development environment (IDE) for the Java programming language, developed mainly for educational purposes, but also suitable for small-scale software development.

BlueJ was developed to support the learning and teaching of object-oriented programming, and its design differs from other development environments as a result. The main screen graphically shows the class structure of an application under development (in a UML-like diagram), and objects can be interactively created and tested. This interaction facility, combined with a clean, simple user interface, allows easy experimentation with objects under development. Object-oriented concepts (classes, objects, communication through method calls) are represented visually and in its interaction design in the interface.

BlueJ is the primary recommended learning software for the Java section of the Computer Application course in ICSE schools all over India, where it is considered the de facto software for learning the basics of Object Oriented Programming and has

proven extremely popular due to its ease of use and wide support in schools and educational centers. However, it is not necessary that code that is written in the actual laboratory tests and final examinations be written in BlueJ, instead any IDE that supports Java can be used.

## **CRYPTOGRAPHY:**

Cryptography is the practice and study of techniques for secure communication in the presence of third parties (called adversaries). More generally, it is about constructing and analyzing protocols that overcome the influence of adversaries and which are related to various aspects in information security such as data confidentiality, data integrity, authentication, and non-repudiation. Modern cryptography intersects the disciplines of mathematics, computer science, and electrical engineering. Applications of cryptography include ATM cards, computer passwords, and electronic commerce.

Cryptography prior to the modern age was effectively synonymous with encryption, the conversion of information from a readable state to apparent nonsense. The originator of an encrypted message shared the decoding technique needed to recover the original information only with intended recipients, thereby precluding unwanted persons to do the same. Since World War I and the advent of the computer, the methods used to carry out cryptology have become increasingly complex and its application more widespread.

Modern cryptography is heavily based on mathematical theory and computer science practice; cryptographic algorithms are designed around computational hardness assumptions, making such algorithms hard to break in practice by any adversary. It is theoretically possible to break such a system, but it is infeasible to do so by any known practical means. These schemes are therefore termed computationally secure, i.e., it would take a relatively large amount of time to obtain the original text even with the use of the most advanced computing machinery.

Symmetric Key Cryptography refers to encryption methods in which both the sender and receiver share and use the same key for encryption as well as decryption processes. Symmetric key ciphers are implemented as either block ciphers or stream ciphers. A block cipher enciphers input in blocks of plain text as opposed to individual characters, the input form used by a stream cipher. This gives us the cipher text. Upon deciphering with the key we get original plain text.

**TERMS USED IN CRYPTOGRAPHY:**

- Cryptography:** Cryptography is the practice and study of techniques for secure communication over public communication mediums in the presence of third parties (called adversaries).
- Cipher:** Ciphers are algorithms for performing encryption or decryption—a series of well-defined steps that can be followed as a procedure.
- Plain Text:** Plain Text is the raw unedited data that a sender wants to transmit to a receiver privately, i.e., without allowing anyone else to have access to the same.
- Cipher Text:** Cipher Text is the result of encryption performed on the Plain Text using a cipher. It is also known as encrypted or encoded information because it contains a form of the original Plain Text that is unreadable without the proper cipher to decrypt it.
- Encryption:** Encryption is the process of encoding messages or information in such a way that only authorized parties can read it. In an encryption scheme, the Plain Text, is encrypted using an encryption algorithm, thereby generating the Cipher Text.
- Decryption:** Decryption is the process of decoding messages or information in such a way that the original Plain Text is obtained. In a decryption scheme the Plain Text, is obtained from the Cipher Text using a decryption algorithm, thereby generating the readable Plain Text.
- Key:** Key is a piece of information (a parameter) that determines the functional output of a cryptographic algorithm or cipher. Without a key, the algorithm would produce no useful result. In Encryption, a key specifies the particular transformation of Plain Text into Cipher Text, or vice versa during Decryption. Keys are important, as ciphers without variable keys can be trivially broken with only the knowledge of the cipher used.
- Cryptosystem:** Cryptosystem is the ordered list of elements of finite possible Plain Texts, finite possible Cipher Texts, finite possible Keys, and the Encryption and Decryption algorithms which correspond to each Key.

# ABOUT THE PROGRAM

## Introduction:

This program demonstrates various ciphers in java. It accepts the Plain Text and Key (wherever applicable) from the user and generates the Cipher Text using various Encryption algorithms. The Plain Text is then obtained from the Cipher Text using the same Key by applying the respective Decryption algorithm. The Ciphers that have been demonstrated in this program are:

1. Rotation Cipher
2. Atbash Cipher
3. Caesar Positive Shift Cipher
4. Caesar Negative Shift Cipher
5. One Time Pad Cipher
6. Monoalphabetic Substitution Cipher
7. Polyalphabetic Substitution Cipher
8. Autokey Cipher
9. Running Key Cipher
10. Polybius Square Cipher
11. Bifid Cipher
12. Trifid Cipher
13. Bitwise Rotation Cipher
14. Bitwise Atbash Cipher
15. Bitwise Binary Complement Cipher
16. Bitwise Left Shift Cipher
17. Bitwise Right Shift Cipher
18. Bitwise Exclusive Or Cipher
19. Bitwise One Time Pad Cipher
20. Square Transposition Cipher
21. Columnar Transposition Cipher
22. Double Transposition Cipher
23. Myszkowski Transposition Cipher
24. Chinese Transposition Cipher
25. Rail Fence Cipher

The algorithms for each of the above Ciphers are described in the next section.



**Menu Driven Program:**

This program is a Menu Driven Program which has a Command Line Interface in the console window. In this a menu with a set of options is displayed to the user in the form of a menu. Each menu item has a unique serial number by which it can be identified. The user is then asked to enter his choice in the form of the serial number of the option he chooses. This then directs the program control to that section of the code.

This program makes the use of nested menus, i.e., menus within other menu items. Indentation is used to show sub-menus, thereby distinguishing between various levels of nesting of menus. If one goes back to the previous menu the indentation level is the same as the level of nesting, giving the user a graphical representation of his location within the menus.

Navigation between the menus is done by choosing the relevant choice in the menu. One can go up one level at a time. All items in the menu do not lead to a sub-menu. Some of them end after printing some data and others after processing data and giving the result to the user. The control then automatically returns to the menu from which the option was invoked. Thus one can traverse only one menu level at a time,

**Data Format Validation;**

This program validates the format of the data that it accepts as input from the user. This means that the data type and the data range if the input is checked to see if it matches with the preset conditions of the algorithm. This prevents processing of the wrong type of data or when the data is out of range, thereby preventing some types of run-time errors.

An example of this is when the user has to enter his choice of the menu item he wants to run in the format of a number. Here it is first checked to see if it is a number. If not then the appropriate message is displayed. If it is fine then it is checked to see if it is within the range of the serial numbers in the menu. If not then the appropriate error message is displayed. If it is accepted then the program goes on to execute the menu item chosen by the user.

Exception Handling is used to catch errors and display the appropriate message to the user. This is mainly used to check if the input is a number or not.

# ALGORITHM OF CIPHERS

## 1. Rotation Cipher:

The Rotation Cipher is implemented as ROT48 ('rotate by 48 places'). It is a simple character substitution cipher that replaces a character with the character 48 places after it in the ASCII table.

For the purpose of this program we consider characters with ASCII values from 32 to 126 which include all the alphabets, numbers and special characters. This set has 95 characters. We take  $95/2 = 47.5 \sim 48$ . Thus we rotate by 48 places according to the ASCII table. If the ASCII value of the result is greater than 126 or less than 32 it is then brought back to the appropriate range, i.e., 32 through 126, using the modulus operator.

For encryption the process is done by rotating it in one direction, i.e., by adding 48 to the ASCII value of the character. This is done for each character in the Plain Text to obtain the Cipher Text.

For decryption the process is done by rotating it in the opposite direction, i.e., by subtracting 48 to the ASCII value of the character. This is done for each character in the Cipher Text to obtain the Plain Text.

### Example:

Plain Text:           This Is The Plain-Text!!

Cipher Text:         %9:DPyDP%96P!=2:~]%6IEQQ

## 2. Atbash Cipher:

Atbash is a simple substitution cipher for the Hebrew alphabet. It consists of substituting aleph (the first letter) for tav (the last), beth (the second) for shin (one before last), and so on, reversing the alphabet. The same algorithm is applied to the ASCII character.

For the purpose of this program we consider characters with ASCII values from 32 to 126 which include all the alphabets, numbers and special characters. This set has 95 characters. Thus we replace the character with its corresponding character in the reverse order of arrangement of the character by their ASCII from 126 to 32.

For encryption the process is done by replacing the character with ASCII value 32 with the character with ASCII value 126, replacing the character with ASCII value 33 with the character with ASCII value 125, replacing the character with ASCII value 34 with the character with ASCII value 124, and so on till replacing the character with ASCII value 126 with the character with ASCII value 32. This is done for each character in the Plain Text to obtain the Cipher Text.

For decryption the same process employed in the encryption is used. Thus it is done by replacing the character with ASCII value 32 with the character with ASCII value 126, replacing the character with ASCII value 33 with the character with ASCII value 125, replacing the character with ASCII value 34 with the character with ASCII value 124, and so on till replacing the character with ASCII value 126 with the character with ASCII value 32. This is done for each character in the Cipher Text to obtain the Plain Text.

### Example:

Plain Text:            This Is The Plain-Text!!

Cipher Text:        J65+~U+~J69~N2=50qJ9&\*} }

### 3. Caesar Positive Shift Cipher:

The Caesar Positive Shift Cipher is implemented as a rotation cipher. It is a simple character substitution cipher that replaces a character with the character 'Key' number of places after it in the ASCII table. The value of 'Key' is taken as input from the user.

For the purpose of this program we consider characters with ASCII values from 32 to 126 which include all the alphabets, numbers and special characters. This set has 95 characters. The numeric value of Key is obtained from the user. Then we shift the character by 'Key' places positively according to the ASCII table. If the ASCII value of the result is greater than 126 or less than 32 it is then brought back to the appropriate range, i.e., 32 through 126, using the modulus operator.

For encryption the rotation process is done by adding 'Key' to the ASCII value of the character. This is done for each character in the Plain Text to obtain the Cipher Text.

For decryption the rotation process is done by subtracting 'Key' from the ASCII value of the character. This is done for each character in the Cipher Text to obtain the Plain Text.

#### Example:

Plain Text:            This Is The Plain-Text!!

Keyword:             3

Cipher Text:        Wklv#Lv#Wkh#Sodlq0Wh{w\$\$

#### 4. Caesar Negative Shift Cipher:

The Caesar Negative Shift Cipher is implemented as a rotation cipher. It is a simple character substitution cipher that replaces a character with the character 'Key' number of places before it in the ASCII table. The value of 'Key' is taken as input from the user.

For the purpose of this program we consider characters with ASCII values from 32 to 126 which include all the alphabets, numbers and special characters. This set has 95 characters. The numeric value of Key is obtained from the user. Then we shift the character by 'Key' places negatively according to the ASCII table. If the ASCII value of the result is greater than 126 or less than 32 it is then brought back to the appropriate range, i.e., 32 through 126, using the modulus operator.

For encryption the rotation process is done by subtracting 'Key' from the ASCII value of the character. This is done for each character in the Plain Text to obtain the Cipher Text.

For decryption the rotation process is done by adding 'Key' to the ASCII value of the character. This is done for each character in the Cipher Text to obtain the Plain Text.

#### Example:

Plain Text:            This Is The Plain-Text!!

Keyword:             3

Cipher Text:        Qefp|Fp|Qeb|Mi^fk\*Qbuq} }

## 5. One Time Pad Cipher:

The One Time Pad Cipher is implemented as a rotation cipher. It is a simple character substitution cipher that replaces a character of the Plain Text by the number of places represented by the corresponding character in the 'Keyword'. For this the length of the 'Keyword' should be at least as long as the Plain Text. This is considered to be an unbreakable Cipher.

For the purpose of this program we consider characters with ASCII values from 32 to 126 which include all the alphabets, numbers and special characters. This set has 95 characters. The 'Keyword' is obtained from the user. Then we shift each character of the Plain Text by the corresponding character of the 'Keyword' places after it according to the ASCII table. If the ASCII value of the result is greater than 126 or less than 32 it is then brought back to the appropriate range, i.e., 32 through 126, using the modulus operator.

For encryption the rotation process is done by adding the relative value of each character in the Plain Text to the relative value of the corresponding character of the 'Keyword' according to their ASCII values. This is done for each character in the Plain Text to obtain the Cipher Text.

For decryption the rotation process is done by subtracting the relative value of each character in the Plain Text to the relative value of the corresponding character of the 'Keyword' according to their ASCII values. This is done for each character in the Plain Text to obtain the Cipher Text.

### Example:

Plain Text:            This Is The Plain-Text!!

Keyword:             Computer Science Is My Favourite Subject.

Cipher Text:         wXWdu>YrT<li6[EO nVHeFn!G

## 6. Monoalphabetic Substitution Cipher:

The Monoalphabetic Substitution Cipher is implemented as a substitution cipher where each character is replaced by another character based on the 'Key'. The 'Key' is prepared from the 'Keyword'.

For the purpose of this program we consider characters with ASCII values from 32 to 126 which include all the alphabets, numbers and special characters. This set has 95 characters. The 'Keyword' is obtained from the user. The 'Key' is then prepared by removing the repeating characters in it and then appending to it the remaining characters with ASCII values from 32 to 126, which do not appear in the 'Keyword'. Thus there are 95 elements in the 'Key' each with its serial number from 32 to 126.

For encryption each character in the Plain Text is taken. The ASCII value of this character is considered as the serial number and the corresponding character from the 'Key' is taken as the Cipher Text. This is done for each character in the Plain Text to obtain the Cipher Text.

For decryption each character in the Cipher Text is taken. This character is compared with all the characters in the 'Key'. When a match is found the corresponding serial number is taken as the character in the Plain Text. This is done for each character in the Cipher Text to obtain the Plain Text.

### Example:

Plain Text:            This Is The Plain-Text!!

Keyword:             Computer

Cipher Text:        MabnCANCMa^CIfZbh%M^xqoo

## 7. Polyalphabetic Substitution Cipher:

The Polyalphabetic Substitution Cipher is implemented as a substitution cipher where each character is replaced by another character which is obtained by shifting the Plain Text character by the corresponding Keyword character.

For the purpose of this program we consider characters with ASCII values from 32 to 126 which include all the alphabets, numbers and special characters. This set has 95 characters. Each character is shifted by the ASCII value of the corresponding character of the Keyword. If the ASCII value of the result is greater than 126 or less than 32 it is then brought back to the appropriate range, i.e., 32 through 126, using the modulus operator.

For encryption each character in the Plain Text is taken. This character is then shifted in one direction by the corresponding character in the Keyword by adding both of them. If the length of the Keyword is less than the length of the Plain Text then it is repeated till it is long enough. This is done for each character in the Plain Text to obtain the Cipher Text.

For decryption each character in the Cipher Text is taken. This character is then shifted in the other direction by the corresponding character in the Keyword by adding both of them. If the length of the Keyword is less than the length of the Cipher Text then it is repeated till it is long enough. This is done for each character in the Cipher Text to obtain the Plain Text.

### Example:

Plain Text:            This Is The Plain-Text!!

Keyword:             Computer

Cipher Text:        wXWdu>YrwXSpFaG\2|BVnifs



## 8. Autokey Cipher:

The Autokey Substitution Cipher is implemented as a substitution cipher where each character is replaced by another character which is obtained by shifting the Plain Text character by the corresponding 'Key' character. The 'Key' is obtained from the 'Keyword' by appending the Plain Text to the end of the Keyword.

For the purpose of this program we consider characters with ASCII values from 32 to 126 which include all the alphabets, numbers and special characters. This set has 95 characters. The 'Keyword' is obtained from the user. The 'Key' is then prepared by appending the Plain Text to the end of the Keyword. Each character is shifted by the ASCII value of the corresponding character of the 'Key'. If the ASCII value of the result is greater than 126 or less than 32 it is then brought back to the appropriate range, i.e., 32 through 126, using the modulus operator.

For encryption each character in the Plain Text is taken. This character is then shifted in one direction by the corresponding character in the Keyword by adding both of them. This is done for each character in the Plain Text to obtain the Cipher Text.

For decryption each character in the Cipher Text is taken. This character is then shifted in the other direction by the corresponding character in the Keyword by adding both of them. This is done for each character in the Cipher Text to obtain the Plain Text.

### Example:

Plain Text:            This Is The Plain-Text!!

Keyword:             Computer

Cipher Text:         wXWdu>Yr)QOsP6UiCu:eIabj

## 9. Running Key Cipher:

The Running Key Cipher is implemented as a substitution cipher where each character is replaced by another character which is obtained by shifting the Plain Text character by the corresponding Keyword character. For this the length of the 'Keyword' should be at least as long as the Plain Text.

For the purpose of this program we consider characters with ASCII values from 32 to 126 which include all the alphabets, numbers and special characters. This set has 95 characters. Each character is shifted by the ASCII value of the corresponding character of the Keyword. If the ASCII value of the result is greater than 126 or less than 32 it is then brought back to the appropriate range, i.e., 32 through 126, using the modulus operator.

For encryption each character in the Plain Text is taken. This character is then shifted in one direction by the corresponding character in the Keyword by adding both of them. This is done for each character in the Plain Text to obtain the Cipher Text.

For decryption each character in the Cipher Text is taken. This character is then shifted in the other direction by the corresponding character in the Keyword by subtracting the character in the Keyword from the character in the Cipher Text. This is done for each character in the Cipher Text to obtain the Plain Text.

### Example:

Plain Text:            This Is The Plain-Text!!

Keyword:             Computer Science Is My Favourite Subject.

Cipher Text:        wXWdu>YrT<Ii6[EO nVHeFn!G

## 10. Polybius Square Cipher:

The Polybius Square Cipher makes use of the Polybius square. The square has 10 rows and 10 columns each numbered from 0 through 9. Each element in the square is a character. It is used to substitute the characters for numbers corresponding to their location in the square in the format 'row number', 'column number'. The Cipher Text generated by the program is a sequence of 3 digit ASCII values of the Cipher Text characters.

For the purpose of this program we consider characters with ASCII values from 32 to 126 which include all the alphabets, numbers and special characters. This set has 95 characters. Thus a Polybius Square of size 10X10 (10 rows and 10 columns) has 100 elements which can accommodate the 95 characters under consideration for the Cipher.

For encryption the character in the plain text is looked up in the Polybius square. Once it is found it is replaced with the pair of numbers 'row number' and 'column number' corresponding to that character. This is done for each character in the Plain Text to obtain the Cipher Text.

For decryption the reverse process employed for encryption is used, i.e., each pair of numbers in the Cipher Text is used to locate the position of the character in the Polybius Square as the first number represents the row number and the second number represents the column number. They are then replaced by the character in that location. This is done for each character in the Plain Text to obtain the Cipher Text.

### Example:

Plain Text:            This Is The Plain-Text!!

Cipher Text:        527273830041830052726900487665737813526988840101

## 11. Bifid Cipher:

The Bifid Cipher makes use of the Polybius square. The square has 10 rows and 10 columns each numbered from 0 through 9. Each element in the square is a character. It is used to substitute the characters for numbers corresponding to their location in the square while writing all the row numbers first followed by all the column numbers. The Cipher Text generated by the program is a sequence of 3 digit ASCII values of the Cipher Text characters.

For the purpose of this program we consider characters with ASCII values from 32 to 126 which include all the alphabets, numbers and special characters. This set has 95 characters. Thus a Polybius Square of size 10 X 10 (10 rows and 10 columns) has 100 elements which can accommodate the 95 characters under consideration for the Cipher.

For encryption the character in the plain text is looked up in the Polybius square. Once it is found it is replaced with the 'row number' on the first line and the 'column number' on the second line, corresponding to that character. This is done for each character in the Plain Text. To obtain the Cipher Text all the numbers in the second line are appended to the end of the first line giving a string of numbers.

For decryption the reverse process employed for encryption is used, i.e., the Cipher Text is divided into two lines with the same number of digits in each line. The first line represents the row and the second line represents the column of the corresponding character. Using the corresponding numbers from the two lines the position of the character in the Polybius Square is identified and replaced with the corresponding character. This is done for all the digits of the Cipher Text to obtain the Plain Text.

### Example:

Plain Text:            This Is The Plain-Text!!

Cipher Text:        577804805760476771568800223301302290865383298411

## 12. Trifid Cipher:

The Trifid Cipher uses a cuboid of size 3 X 6 X 6. It can be viewed as 3 squares (numbered from 0 through 2) with 6 rows (numbered from 0 through 5) and 6 columns (numbered from 0 through 5). Each element in the cuboid is a character. It is used to substitute the characters for numbers corresponding to their location in the cuboid by writing all the square numbers first, all the row numbers second followed by all the column numbers third. The Cipher Text generated by the program is a sequence of 3 digit ASCII values of the Cipher Text characters.

For the purpose of this program we consider characters with ASCII values from 32 to 126 which include all the alphabets, numbers and special characters. This set has 95 characters. Thus the cuboid of size 3 X 6 X 6 (3 squares, 6 rows and 6 columns) has 108 elements which can accommodate the 95 characters under consideration for the Cipher.

For encryption the character in the plain text is looked up in the cuboid. Once it is found it is replaced with the 'square number' on the first line, 'row number' on the second line and the 'column number' on the third line, corresponding to that character. This is done for each character in the Plain Text. To obtain the Cipher Text all the numbers in the third line are appended to the second line and this result is appended to the end of the first line giving a string of numbers.

For decryption the reverse process employed for encryption is used, i.e., the Cipher Text is divided into three lines with equal number of digits in each line. The first line represents the square, the second line represents the row and the third line represents the column of the corresponding character. Using the corresponding numbers from the three lines the position of the character in the cuboid is identified and replaced with the corresponding character. This is done for all the digits of the Cipher Text to obtain the Plain Text.

### Example:

Plain Text:            This Is The Plain-Text!!

Cipher Text:        12220120121012122011220020010010205020401225220040150  
                         5504030045101434011

### 13. Bitwise Rotation Cipher:

In the Bitwise Rotation Cipher the rotation cipher is applied at the bit level. All the ASCII characters can be represented by 8 bits (1 byte). This gives us  $2^8 = 256$  possible characters. The Cipher Text generated by the program is a sequence of 3 digit ASCII values of the Cipher Text characters.

For encryption each character in the Plain Text is converted to its representation in the 8 bit form. It is then shuffled as per the following table. The rearranged bits then represent the character. This is done for all the characters to obtain the Cipher Text.

For decryption each character in the Cipher Text is converted to its representation in the 8 bit form. It is then shuffled as per the following table. The rearranged bits then represent the character. This is done for all the characters to obtain the Plain Text.

Bit Position In Byte	Bit Position (Before)	Bit Position (After)
0	0	4
1	1	5
2	2	6
3	3	7
4	4	0
5	5	1
6	6	2
7	7	3

#### Example:

Plain Text: This Is The Plain-Text!!

Cipher Text: 06913415005500214805500206913408600200519802215023021  
0069086135071018018

#### 14. Bitwise Atbash Cipher:

In the Bitwise Atbash Cipher the atbash cipher is applied at the bit level. All the ASCII characters can be represented by 8 bits (1 byte). This gives us  $2^8 = 256$  possible characters. The Cipher Text generated by the program is a sequence of 3 digit ASCII values of the Cipher Text characters.

For encryption each character in the Plain Text is converted to its representation in the 8 bit form. It is then shuffled as per the atbash cipher as shown in the following table. The rearranged bits then represent the character. This is done for all the characters to obtain the Cipher Text.

For decryption each character in the Cipher Text is converted to its representation in the 8 bit form. It is then shuffled as per the atbash cipher as shown in the following table. The rearranged bits then represent the character. This is done for all the characters to obtain the Plain Text.

Bit Position In Byte	Bit Position (Before)	Bit Position (After)
0	0	7
1	1	6
2	2	5
3	3	4
4	4	3
5	5	2
6	6	1
7	7	0

#### Example:

Plain Text: This Is The Plain-Text!!

Cipher Text: 04202215020600414620600404202216600401005413415011818  
0042166030046132132

**15. Bitwise Binary Compliment Cipher:**

In the Bitwise Binary Compliment Cipher each bit is complimented, i.e., 1 becomes 0 and 0 becomes 1. All the ASCII characters can be represented by 8 bits (1 byte). This gives us  $2^8 = 256$  possible characters. The Cipher Text generated by the program is a sequence of 3 digit ASCII values of the Cipher Text characters.

For encryption each character in the Plain Text is converted to its representation in the 8 bit form. Each bit is then complimented using the Logical NOT operation as shown in the following table. The complimented bits then represent the character. This is done for all the characters to obtain the Cipher Text.

For decryption each character in the Cipher Text is converted to its representation in the 8 bit form. Each bit is then complimented using the Logical NOT operation as shown in the following table. The complimented bits then represent the character. This is done for all the characters to obtain the Plain Text.

Bit A	Compliment of Bit A (!A)
0	1
1	0

**Example:**

Plain Text:           This Is The Plain-Text!!

Cipher Text:        17115115014022318214022317115115422317514715815014521  
                          0171154135139222222

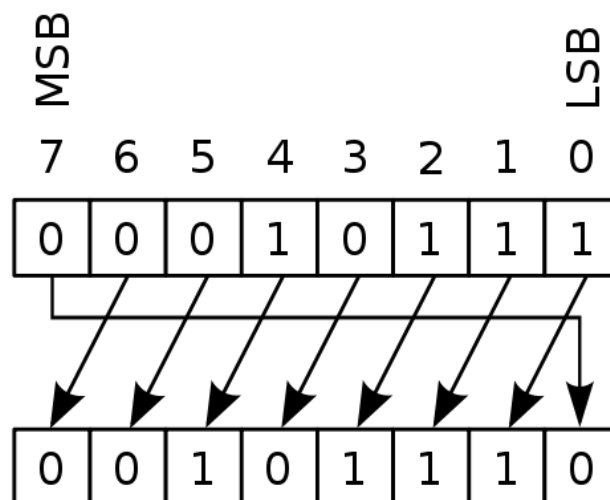


## 16. Bitwise Left Shift Cipher:

In the Bitwise Left Shift Cipher each bit is shifted to the left by the number in the Keyword (0-7). All the ASCII characters can be represented by 8 bits (1 byte). This gives us  $2^8 = 256$  possible characters. The Cipher Text generated by the program is a sequence of 3 digit ASCII values of the Cipher Text characters.

For encryption each character in the Plain Text is converted to its representation in the 8 bit form. It is then shifted to the left by the number in the Keyword. The rearranged bits then represent the character. This is done for all the characters to obtain the Cipher Text.

For decryption each character in the Cipher Text is converted to its representation in the 8 bit form. It is then shifted to the right by the number in the Keyword. The rearranged bits then represent the character. This is done for all the characters to obtain the Plain Text.



Bitwise Left Shift By 1 Bit

### Example:

Plain Text: This Is The Plain-Text!!

Keyword: 3

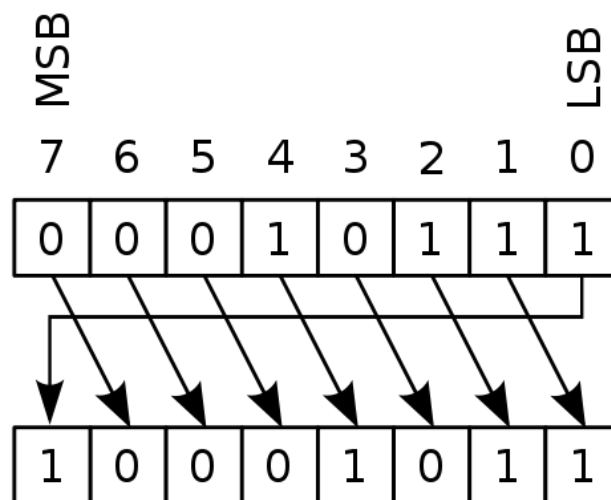
Cipher Text: 16206707515500107415500116206704300113009901107511510  
5162043195163009009

### 17. Bitwise Right Shift Cipher:

In the Bitwise Right Shift Cipher each bit is shifted to the right by the number in the Keyword (0-7). All the ASCII characters can be represented by 8 bits (1 byte). This gives us  $2^8 = 256$  possible characters. The Cipher Text generated by the program is a sequence of 3 digit ASCII values of the Cipher Text characters.

For encryption each character in the Plain Text is converted to its representation in the 8 bit form. It is then shifted to the right by the number in the Keyword. The rearranged bits then represent the character. This is done for all the characters to obtain the Cipher Text.

For decryption each character in the Cipher Text is converted to its representation in the 8 bit form. It is then shifted to the left by the number in the Keyword. The rearranged bits then represent the character. This is done for all the characters to obtain the Plain Text.



Bitwise Right Shift By 1 Bit

#### Example:

Plain Text:            This Is The Plain-Text!!

Keyword:                3

Cipher Text:            13801304511000404111000413801317200401014104404520516  
5138172015142036036

## 18. Bitwise Exclusive Or Cipher:

In the Bitwise Exclusive Or Cipher the logical XOR (Exclusive Or) operator is used. The corresponding bits of the Keyword and the Plain Text on applying the XOR operation give the Cipher Text. All the ASCII characters can be represented by 8 bits (1 byte). This gives us  $2^8 = 256$  possible characters. The Cipher Text generated by the program is a sequence of 3 digit ASCII values of the Cipher Text characters.

For encryption each character in the Plain Text is converted to its representation in the 8 bit form. Also each character in the Keyword is converted to its representation in the 8 bit form. If the length of the Keyword is lesser than the length of the Plain Text it is repeated till it is as long as Plain Text. The XOR operation is applied to the corresponding bits of the Keyword and the Plain Text as per the following table. These bits then represent the corresponding character in the Cipher Text. This is done for all the characters to obtain the Cipher Text.

For decryption each character in the Cipher Text is converted to its representation in the 8 bit form. Also each character in the Keyword is converted to its representation in the 8 bit form. If the length of the Keyword is lesser than the length of the Cipher Text it is repeated till it is as long as Cipher Text. The XOR operation is applied to the corresponding bits of the Keyword and the Cipher Text as per the following table. These bits then represent the corresponding character in the Plain Text. This is done for all the characters to obtain the Plain Text.

Bit A (Keyword)	Bit B (PT/CT)	A XOR B (CT/PT)
0	0	0
0	1	1
1	0	1
1	1	0

### Example:

Plain Text: This Is The Plain-Text!!

Keyword: Computer

Cipher Text: 02300700400308506102208202300700808003702400402704506  
6057021013000068083

**19. Bitwise One Time Pad Cipher:**

In the Bitwise One Time Pad Cipher the logical XOR (Exclusive Or) operator is used. The corresponding bits of the Keyword and the Plain Text on applying the XOR operation give the Cipher Text. The Keyword must be at least as long as the Plain Text. All the ASCII characters can be represented by 8 bits (1 byte). This gives us  $2^8 = 256$  possible characters. The Cipher Text generated by the program is a sequence of 3 digit ASCII values of the Cipher Text characters.

For encryption each character in the Plain Text is converted to its representation in the 8 bit form. Also each character in the Keyword is converted to its representation in the 8 bit form. The XOR operation is applied to the corresponding bits of the Keyword and the Plain Text as per the following table. These bits then represent the corresponding character in the Cipher Text. This is done for all the characters to obtain the Cipher Text.

For decryption each character in the Cipher Text is converted to its representation in the 8 bit form. Also each character in the Keyword is converted to its representation in the 8 bit form. The XOR operation is applied to the corresponding bits of the Keyword and the Cipher Text as per the following table. These bits then represent the corresponding character in the Plain Text. This is done for all the characters to obtain the Plain Text.

Bit A (Keyword)	Bit B (PT/CT)	A XOR B (CT/PT)
0	0	0
0	1	1
1	0	1
1	1	0

**Example:**

Plain Text: This Is The Plain-Text!!

Keyword: Computer Science Is My Favourite Subject.

Cipher Text: 02300700400308506102208211605900607305300200201207810  
0039069053013001103

**20. Square Transposition Cipher:**

The Square Transposition cipher arranges the Plain Text in a perfect square, i.e., number of rows equal the number of columns, where each element in the square has a character. If there are empty elements they are padded with spaces. The cipher is obtained by interchanging the rows with the columns.

For encryption the minimum size of the square is calculated based on the size of the Plain Text. The square is then filled up with the Plain Text row wise. If there are any empty elements left they are filled with spaces. The Cipher Text is obtained by reading the square column wise from left to right.

For decryption the minimum size of the square is calculated based on the size of the Cipher Text. The square is then filled up with the Cipher Text column wise. If there are any empty elements left they are filled with spaces. The Cipher Text is obtained by reading the square row wise from top to bottom.

**Example:**

Plain Text:            This Is The Plain-Text!!

Cipher Text:        T!eixhs nti P-!sTIT! hae

## 21. Columnar Transposition Cipher:

In a columnar transposition, the message is written out in rows of a fixed length, and then read out again column by column, and the columns are chosen in some scrambled order. Both the width of the rows and the permutation of the columns are based on the Keyword.

The length of the Keyword determines the number of columns. The order of the characters, as determined by the ASCII values of the characters, determines the order of reading of the columns. If two or more characters of the Keyword are the same then preference is given to the column on the left.

For encryption the Plain Text is written out in rows of length equal to the length of the Keyword. If there are any empty elements left they are filled with spaces to complete the row. Thus the number of columns equals the length of the Keyword. The Cipher Text is obtained by reading out the columns in the order of the corresponding characters of the keywords, as determined by the ASCII values of the characters.

For decryption the Cipher Text is filled into columns of length equal to the length of the Plain Text divided by the length of the Keyword in the order of the corresponding characters of the keywords, as determined by the ASCII values of the characters. If there are any empty elements left they are filled with spaces to complete the column. The Plain Text is obtained by reading out the rows in order from top to bottom.

### Example:

Plain Text:            This Is The Plain-Text!!

Keyword:             Computer

Cipher Text:         TTnsa!ieThh-s e i!It Px

**22. Double Transposition Cipher:**

The Double Transposition Cipher is simply the Columnar Transposition Cipher applied twice. The same Keyword can be used for both transpositions or two different Keywords can be used.

The length of the Keyword in each stage determines the number of columns in use. The order of the characters, as determined by the ASCII values of the characters, determines the order of reading of the columns. If two or more characters of the Keyword are the same then preference is given to the column on the left.

**Example:**

Plain Text:	This Is The Plain-Text!!
Keyword 1:	Computer
Keyword 2:	Project
Cipher Text:	Tee !sl a-I sh! nhixTT Pi t

### 23. Myszkowski Transposition Cipher:

The Myszkowski Transposition is a variant form of Columnar Transposition, proposed by Émile Victor Théodore Myszkowski in 1902. It requires a keyword with recurrent letters. In Columnar Transposition subsequent occurrences of a keyword letter are treated as if the next letter in alphabetical order, e.g., the keyword TOMATO yields a numeric keystream of '532164'. In Myszkowski Transposition, recurrent keyword letters are numbered identically, thus TOMATO yielding a keystream of '432143'.

The length of the Keyword determines the number of columns. The order of the characters, as determined by the ASCII values of the characters, determines the order of reading of the columns. If two or more characters of the Keyword are the same then they are read from left to right within each row.

For encryption the Plain Text is written out in rows of length equal to the length of the Keyword. If there are any empty elements left they are filled with spaces to complete the row. Thus the number of columns equals the length of the Keyword. The Cipher Text is obtained by reading out the columns with unique numbers downwards while for those with recurring numbers are read left to right within the row.

For decryption the Cipher Text is filled into columns of length equal to the length of the Plain Text divided by the length of the Keyword in the order as described in the previous paragraph. If there are any empty elements left they are filled with spaces. The Plain Text is obtained by reading out the rows in order from top to bottom.

#### Example:

Plain Text:            This Is The Plain-Text!!

Keyword:             Computer

Cipher Text:         TTnsa!ieThh-s e i!lt Px



## 24. Chinese Transposition Cipher:

In the Chinese Transposition Cipher, the letters of the message are written from right to left, down and up columns to scramble the letters. Then, starting in the first row, the letters are taken in order to get the new Cipher Text. The Keyword determined the number of rows. All remaining blanks are filled with spaces.

For example, if the message needed to be enciphered was ‘THE DOG RAN FAR.’ and the Keyword was 4, the Chinese cipher would look like this:

.	R		T
R	A	G	H
A	N	O	E
F		D	

The Cipher Text is obtained by reading the table row wise from top to bottom. This reads as ‘.R TRAGHANOEF DE’.

The Plain Text is obtained by writing the Cipher Text in rows of equal length, where the number of rows is determined by the Keyword, and then reading it from right to left, down and up the columns.

### Example:

Plain Text: This Is The Plain-Text!!

Keyword: 3

Cipher Text: !T-P sIT!enle htxiahTsi

**25. Rail Fence Cipher:**

The Rail Fence Cipher is a form of transposition cipher that gets its name from the way in which it is encoded. In the rail fence cipher, the Plain Text is written downwards on successive ‘rails’ of an imaginary fence, then moving up when we get to the bottom. Blanks are filled with spaces. The message is then read off in rows from left to right.

For example, using 4 ‘rails’ and a message of ‘THE DOG RAN.’, writes out as:

T	-	-	-	-	-	G	-	-	-	-	-
-	H	-	-	-	O	-	-	-	-	.	-
-	-	E	-	D	-	-	-	R	-	N	-
-	-	-	-	-	-	-	-	-	A	-	-

The Cipher Text obtained by reading the rows from top to bottom from the above table is: ‘TG HO .EDRN A’

The Plain Text is obtained by arranging the Cipher Text into rows and then reading it in the rail fence manner.

**Example:**

Plain Text: This Is The Plain-Text!!

Keyword: 3

Cipher Text: T TPnx hsI h li-et! iseaT!

# SYSTEM REQUIREMENT

## HARDWARE:

1. CPU: Pentium® 4 CPU 2.00GHz
2. RAM: 2.02GHz. 0.99GB of RAM
3. HDD Samsung 80GB
4. Monitor: Dell LCD Monitor
5. Mouse: UMax Optical USB Mouse
6. Keyboard: Logitech Keyboard
7. Speaker: ZenStar Speakers

## SOFTWARE:

1. Operating System: Microsoft Windows XP
2. Integrated Development Environment (IDE): BlueJ

# SOURCE CODE

```
/*  
* ICSE Std. 10 Computer Science Project (Java)  
*  
* Title: Implementation Of Cryptography In Java  
*  
* Number of Lines of Code: 10,160  
*  
*  
*  
* Summary:  
*  
* This Program Is A Menu Driven Program Demonstrating  
* The Implementation of Various Ciphers In Java. This  
* Program Also Includes Code For Validating The Input  
* Format Received From The User.  
*  
* The Following Ciphers Have Been Implemented In Java  
*  
* 1. Rotation Cipher  
* 2. Atbash Cipher  
* 3. Caesar Positive Shift Cipher  
* 4. Caesar Negative Shift Cipher  
* 5. One Time Pad Cipher  
* 6. Monoalphabetic Substitution Cipher  
* 7. Polyalphabetic Substitution Cipher  
* 8. Autokey Cipher  
* 9. Running Key Cipher  
* 10. Polybius Square Cipher  
* 11. Bifid Cipher  
* 12. Trifid Cipher  
* 13. Bitwise Rotation Cipher  
* 14. Bitwise Atbash Cipher  
* 15. Bitwise Binary Compliment Cipher  
* 16. Bitwise Left Shift Cipher  
* 17. Bitwise Right Shift Cipher  
* 18. Bitwise Exclusive Or Cipher  
* 19. Bitwise One Time Pad Cipher  
* 20. Square Transposition Cipher  
* 21. Columnar Transposition Cipher  
* 22. Double Transposition Cipher  
* 23. Myszowski Transposition Cipher  
* 24. Chinese Transposition Cipher  
* 25. Rail Fence Cipher  
*  
*  
*  
* Author Details:  
*  
* Name: Omkar S. Nath
```

```
* Class: 10-D
* Roll Number: 35
* Computer Code: 230438
*
* School: The Bishop's School, Camp, Pune
*/
```

```
import java.io.*;
```

```
class Cryptography
```

```
{
    public static int indent=1;

    public static void main()throws IOException
    {
        BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

        int f1;
        int f2;
        int f3;
        int ch1;
        int ch2;
        int ch3;
        int temp;

        ch1=0;
        f1=0;
        do
        {
            temp=0;
            while(temp==0)
            {
                System.out.println();
                System.out.println("HOME: IMPLEMENTATION OF CRYPTOGRAPHY IN JAVA");
                System.out.println("1. About");
                System.out.println("2. Ciphers");
                System.out.println("3. Author");
                System.out.println("4. Exit");
                System.out.print("Enter Choice: ");

                try
                {
                    ch1=Integer.parseInt(bfrd.readLine());
                    temp=1;
                }
                catch(NumberFormatException E1nfe)
                {
                    System.out.println("Choice Should Be A Number");
                    temp=0;
                }
            }

            switch(ch1)
            {
                case 1:
```

```

f1=0;
if(indent==1)System.out.print("\t");
System.out.println();
if(indent==1)System.out.print("\t");
System.out.println("HOME.1. ABOUT: IMPLEMENTATION OF CRYPTOGRAPHY IN
JAVA");
if(indent==1)System.out.print("\t");
System.out.println("");
if(indent==1)System.out.print("\t");
System.out.println("Cryptography is the practice and study of techniques");
if(indent==1)System.out.print("\t");
System.out.println("for secure communication in the presence of third");
if(indent==1)System.out.print("\t");
System.out.println("parties (called adversaries). This program demonstrates");
if(indent==1)System.out.print("\t");
System.out.println("various ciphers in java. It accepts the Plain Text and");
if(indent==1)System.out.print("\t");
System.out.println("Key (wherever applicable) from the user and generates");
if(indent==1)System.out.print("\t");
System.out.println("the Cipher Text using various Encryption algorithms. The ");
if(indent==1)System.out.print("\t");
System.out.println("Plain Text is then obtained from the Cipher Text using");
if(indent==1)System.out.print("\t");
System.out.println("the same Key by applying the respective Decryption");
if(indent==1)System.out.print("\t");
System.out.println("algorithm.");
break;

```

case 2:

```

f1=0;

ch2=0;
f2=0;
do
{
    temp=0;
    while(temp==0)
    {
        if(indent==1)System.out.print("\t");
        System.out.println();
        if(indent==1)System.out.print("\t");
        System.out.println("HOME.2. CIPHERS");
        if(indent==1)System.out.print("\t");
        System.out.println("1. Rotation Cipher");
        if(indent==1)System.out.print("\t");
        System.out.println("2. Atbash Cipher");
        if(indent==1)System.out.print("\t");
        System.out.println("3. Caesar Positive Shift Cipher");
        if(indent==1)System.out.print("\t");
        System.out.println("4. Caesar Negative Shift Cipher");
        if(indent==1)System.out.print("\t");
        System.out.println("5. One Time Pad Cipher");
        if(indent==1)System.out.print("\t");
        System.out.println("6. Monoalphabetic Substitution Cipher");
        if(indent==1)System.out.print("\t");
    }
}

```

```
System.out.println("7. Polyalphabetic Substitution Cipher");
if(indent==1)System.out.print("\t");
System.out.println("8. Autokey Cipher");
if(indent==1)System.out.print("\t");
System.out.println("9. Running Key Cipher");
if(indent==1)System.out.print("\t");
System.out.println("10. Polybius Square Cipher");
if(indent==1)System.out.print("\t");
System.out.println("11. Bifid Cipher");
if(indent==1)System.out.print("\t");
System.out.println("12. Trifid Cipher");
if(indent==1)System.out.print("\t");
System.out.println("13. Bitwise Rotation Cipher");
if(indent==1)System.out.print("\t");
System.out.println("14. Bitwise Atbash Cipher");
if(indent==1)System.out.print("\t");
System.out.println("15. Bitwise Binary Compliment Cipher");
if(indent==1)System.out.print("\t");
System.out.println("16. Bitwise Left Shift Cipher");
if(indent==1)System.out.print("\t");
System.out.println("17. Bitwise Right Shift Cipher");
if(indent==1)System.out.print("\t");
System.out.println("18. Bitwise Exclusive Or Cipher");
if(indent==1)System.out.print("\t");
System.out.println("19. Bitwise One Time Pad Cipher");
if(indent==1)System.out.print("\t");
System.out.println("20. Square Transposition Cipher");
if(indent==1)System.out.print("\t");
System.out.println("21. Columnar Transposition Cipher");
if(indent==1)System.out.print("\t");
System.out.println("22. Double Transposition Cipher");
if(indent==1)System.out.print("\t");
System.out.println("23. Myszkowski Transposition Cipher");
if(indent==1)System.out.print("\t");
System.out.println("24. Chinese Transposition Cipher");
if(indent==1)System.out.print("\t");
System.out.println("25. Rail Fence Cipher");
if(indent==1)System.out.print("\t");
System.out.println("26. Back To HOME");
if(indent==1)System.out.print("\t");
System.out.print("Enter Choice: ");

try
{
    ch2=Integer.parseInt(bfrd.readLine());
    temp=1;
}
catch(NumberFormatException E1nfe)
{
    if(indent==1)System.out.print("\t");
    System.out.println("Choice Should Be A Number");
    temp=0;
}
}
```

```
switch(ch2)
{
    case 1:
        f2=0;

        ch3=0;
        f3=0;
        do
        {
            temp=0;
            while(temp==0)
            {
                if(indent==1)System.out.print("\t\t");
                System.out.println();
                if(indent==1)System.out.print("\t\t");
                System.out.println("HOME.2.1. ROTATION CIPHER");
                if(indent==1)System.out.print("\t\t");
                System.out.println("1. About");
                if(indent==1)System.out.print("\t\t");
                System.out.println("2. Encrypt");
                if(indent==1)System.out.print("\t\t");
                System.out.println("3. Decrypt");
                if(indent==1)System.out.print("\t\t");
                System.out.println("4. Back To CIPHERS");
                if(indent==1)System.out.print("\t\t");
                System.out.print("Enter Choice: ");

                try
                {
                    ch3=Integer.parseInt(bfrd.readLine());
                    temp=1;
                }
                catch(NumberFormatException E1nfe)
                {
                    if(indent==1)System.out.print("\t\t");
                    System.out.println("Choice Should Be A Number");
                    temp=0;
                }
            }
        }

        switch(ch3)
        {
            case 1:
                f3=0;
                if(indent==1)System.out.print("\t\t\t");
                System.out.println();
                if(indent==1)System.out.print("\t\t\t");
                System.out.println("HOME.2.1.1. ABOUT: ROTATION CIPHER");
                if(indent==1)System.out.print("\t\t\t");
                System.out.println("");
                if(indent==1)System.out.print("\t\t\t");
                System.out.println("The Rotation Cipher is implemented as ROT48");
                if(indent==1)System.out.print("\t\t\t");
                System.out.println("('rotate by 48 places'). It is a simple");
                if(indent==1)System.out.print("\t\t\t");
```



```
        System.out.println("character substitution cipher that replaces");
        if(indent==1)System.out.print("\t\t");
        System.out.println("a character with the character 48 places");
        if(indent==1)System.out.print("\t\t");
        System.out.println("after it in the ASCII table.");
        break;

    case 2:
        f3=0;
        if(indent==1)System.out.print("\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t");
        System.out.println("HOME.2.1.2. ENCRYPTION: ROTATION CIPHER");
        Rotation_enc();
        break;

    case 3:
        f3=0;
        if(indent==1)System.out.print("\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t");
        System.out.println("HOME.2.1.3. DECRYPTION: ROTATION CIPHER");
        Rotation_dec();
        break;

    case 4:
        f3=1;
        break;

    default:
        f3=0;
        if(indent==1)System.out.print("\t\t");
        System.out.println("Invalid Choice");
    }
}while(f3==0);

break;

case 2:
    f2=0;

    ch3=0;
    f3=0;
    do
    {
        temp=0;
        while(temp==0)
        {
            if(indent==1)System.out.print("\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t");
            System.out.println("HOME.2.2. ATBASH CIPHER");
            if(indent==1)System.out.print("\t\t");
            System.out.println("1. About");
            if(indent==1)System.out.print("\t\t");
```

```
System.out.println("2. Encrypt");
if(indent==1)System.out.print("\t\t");
System.out.println("3. Decrypt");
if(indent==1)System.out.print("\t\t");
System.out.println("4. Back To CIPHERS");
if(indent==1)System.out.print("\t\t");
System.out.print("Enter Choice: ");

try
{
    ch3=Integer.parseInt(bfrd.readLine());
    temp=1;
}
catch(NumberFormatException E1nfe)
{
    if(indent==1)System.out.print("\t\t");
    System.out.println("Choice Should Be A Number");
    temp=0;
}
}

switch(ch3)
{
    case 1:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.2.1. ABOUT: ATBASH CIPHER");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Atbash is a simple substitution cipher");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("for the Hebrew alphabet. It consists");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("of substituting aleph (the first letter)");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("for tav (the last), beth (the second)");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("for shin (one before last), and so on,");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("reversing the alphabet. The same");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("algorithm is applied to the ASCII");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("character.");
        break;

    case 2:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.2.2. ENCRYPTION: ATBASH CIPHER");
```

```
        Atbash_enc();
        break;

    case 3:
        f3=0;
        if(indent==1)System.out.print("\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t");
        System.out.println("HOME.2.2.3. DECRYPTION: ATBASH CIPHER");
        Atbash_dec();
        break;

    case 4:
        f3=1;
        break;

    default:
        f3=0;
        if(indent==1)System.out.print("\t\t");
        System.out.println("Invalid Choice");
    }
    }while(f3==0);

    break;

case 3:
    f2=0;

    ch3=0;
    f3=0;
    do
    {
        temp=0;
        while(temp==0)
        {
            if(indent==1)System.out.print("\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t");
            System.out.println("HOME.2.3. CAESAR POSITIVE SHIFT CIPHER");
            if(indent==1)System.out.print("\t\t");
            System.out.println("1. About");
            if(indent==1)System.out.print("\t\t");
            System.out.println("2. Encrypt");
            if(indent==1)System.out.print("\t\t");
            System.out.println("3. Decrypt");
            if(indent==1)System.out.print("\t\t");
            System.out.println("4. Back To CIPHERS");
            if(indent==1)System.out.print("\t\t");
            System.out.print("Enter Choice: ");

            try
            {
                ch3=Integer.parseInt(bfrd.readLine());
                temp=1;
            }
        }
    }
}
```

```

        catch(NumberFormatException E1nfe)
        {
            if(indent==1)System.out.print("\t\t");
            System.out.println("Choice Should Be A Number");
            temp=0;
        }
    }

    switch(ch3)
    {
        case 1:
            f3=0;
            if(indent==1)System.out.print("\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t");
            System.out.println("HOME.2.3.1. ABOUT: CAESAR POSITIVE SHIFT
CIPHER");

            if(indent==1)System.out.print("\t\t");
            System.out.println("");
            if(indent==1)System.out.print("\t\t");
            System.out.println("The Caesar Positive Shift Cipher is implemented");
            if(indent==1)System.out.print("\t\t");
            System.out.println("as a rotation cipher. It is a simple character");
            if(indent==1)System.out.print("\t\t");
            System.out.println("substitution cipher that replaces a character");
            if(indent==1)System.out.print("\t\t");
            System.out.println("with the character 'Key' number of places after");
            if(indent==1)System.out.print("\t\t");
            System.out.println("it in the ASCII table. The value of 'Key' is taken");
            if(indent==1)System.out.print("\t\t");
            System.out.println("as input from the user.");
            break;

        case 2:
            f3=0;
            if(indent==1)System.out.print("\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t");
            System.out.println("HOME.2.3.2. ENCRYPTION: CAESAR POSITIVE SHIFT
CIPHER");

            CaesarPositiveShift_enc();
            break;

        case 3:
            f3=0;
            if(indent==1)System.out.print("\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t");
            System.out.println("HOME.2.3.3. DECRYPTION: CAESAR POSITIVE SHIFT
CIPHER");

            CaesarPositiveShift_dec();
            break;

        case 4:
            f3=1;

```

```
        break;

        default:
            f3=0;
            if(indent==1)System.out.print("\t\t");
            System.out.println("Invalid Choice");
        }
    }while(f3==0);

    break;

case 4:
    f2=0;

    ch3=0;
    f3=0;
    do
    {
        temp=0;
        while(temp==0)
        {
            if(indent==1)System.out.print("\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t");
            System.out.println("HOME.2.4. CAESAR NEGATIVE SHIFT CIPHER");
            if(indent==1)System.out.print("\t\t");
            System.out.println("1. About");
            if(indent==1)System.out.print("\t\t");
            System.out.println("2. Encrypt");
            if(indent==1)System.out.print("\t\t");
            System.out.println("3. Decrypt");
            if(indent==1)System.out.print("\t\t");
            System.out.println("4. Back To CIPHERS");
            if(indent==1)System.out.print("\t\t");
            System.out.print("Enter Choice: ");

            try
            {
                ch3=Integer.parseInt(bfrd.readLine());
                temp=1;
            }
            catch(NumberFormatException E1nfe)
            {
                if(indent==1)System.out.print("\t\t");
                System.out.println("Choice Should Be A Number");
                temp=0;
            }
        }
    }

    switch(ch3)
    {
        case 1:
            f3=0;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
```

```

        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.4.1. ABOUT: CAESAR NEGATIVE SHIFT
CIPHER");

        if(indent==1)System.out.print("\t\t\t");
        System.out.println("");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("The Caesar Negative Shift Cipher is implemented");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("as a rotation cipher. It is a simple character");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("substitution cipher that replaces a character");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("with the character 'Key' number of places before");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("it in the ASCII table. The value of 'Key' is taken");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("as input from the user.");
        break;

    case 2:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.4.2. ENCRYPTION: CAESAR NEGATIVE
SHIFT CIPHER");

        CaesarNegativeShift_enc();
        break;

    case 3:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.4.3. DECRYPTION: CAESAR NEGATIVE
SHIFT CIPHER");

        CaesarNegativeShift_dec();
        break;

    case 4:
        f3=1;
        break;

    default:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Choice");
    }
}while(f3==0);

break;

case 5:
    f2=0;

```

```
ch3=0;
f3=0;
do
{
    temp=0;
    while(temp==0)
    {
        if(indent==1)System.out.print("\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t");
        System.out.println("HOME.2.5. ONE TIME PAD CIPHER");
        if(indent==1)System.out.print("\t\t");
        System.out.println("1. About");
        if(indent==1)System.out.print("\t\t");
        System.out.println("2. Encrypt");
        if(indent==1)System.out.print("\t\t");
        System.out.println("3. Decrypt");
        if(indent==1)System.out.print("\t\t");
        System.out.println("4. Back To CIPHERS");
        if(indent==1)System.out.print("\t\t");
        System.out.print("Enter Choice: ");

        try
        {
            ch3=Integer.parseInt(bfrd.readLine());
            temp=1;
        }
        catch(NumberFormatException E1nfe)
        {
            if(indent==1)System.out.print("\t\t");
            System.out.println("Choice Should Be A Number");
            temp=0;
        }
    }

    switch(ch3)
    {
        case 1:
            f3=0;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("HOME.2.5.1. ABOUT: ONE TIME PAD CIPHER");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("The One Time Pad Cipher is implemented");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("as a rotation cipher. It is a simple");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("character substitution cipher that");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("replaces a character of the Plain");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Text by the number of places");
```

```
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("represented by the corresponding");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("character in the 'Keyword'. For this");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("the length of the 'Keyword' should be");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("at least as long as the Plain Text.");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("This is considered to be an unbreakable");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Cipher.");
        break;

    case 2:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.5.2. ENCRYPTION: ONE TIME PAD CIPHER");
        OneTimePad_enc();
        break;

    case 3:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.5.3. DECRYPTION: ONE TIME PAD CIPHER");
        OneTimePad_dec();
        break;

    case 4:
        f3=1;
        break;

    default:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Choice");
    }
}while(f3==0);

break;

case 6:
    f2=0;

    ch3=0;
    f3=0;
    do
    {
        temp=0;
        while(temp==0)
        {
```



```

        if(indent==1)System.out.print("\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t");
        System.out.println("HOME.2.6. MONOALPHABETIC SUBSTITUTION
CIPHER");

        if(indent==1)System.out.print("\t\t");
        System.out.println("1. About");
        if(indent==1)System.out.print("\t\t");
        System.out.println("2. Encrypt");
        if(indent==1)System.out.print("\t\t");
        System.out.println("3. Decrypt");
        if(indent==1)System.out.print("\t\t");
        System.out.println("4. Back To CIPHERS");
        if(indent==1)System.out.print("\t\t");
        System.out.print("Enter Choice: ");

        try
        {
            ch3=Integer.parseInt(bfrd.readLine());
            temp=1;
        }
        catch(NumberFormatException E1nfe)
        {
            if(indent==1)System.out.print("\t\t");
            System.out.println("Choice Should Be A Number");
            temp=0;
        }
    }

    switch(ch3)
    {
        case 1:
            f3=0;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("HOME.2.6.1. ABOUT: MONOALPHABETIC
SUBSTITUTION CIPHER");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("The Monoalphabetic Substitution Cipher is implemented");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("as a substitution cipher where each character is");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("replaced by another character based on the ‘Key’. The");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("‘Key’ is prepared from the ‘Keyword’.");
            System.out.println("");
            break;

        case 2:
            f3=0;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();

```

```

        if(indent==1)System.out.print("\t\t");
        System.out.println("HOME.2.6.2. ENCRYPTION: MONOALPHABETIC
SUBSTITUTION CIPHER");
        MonoalphabeticSubstitution_enc();
        break;

    case 3:
        f3=0;
        if(indent==1)System.out.print("\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t");
        System.out.println("HOME.2.6.3. DECRYPTION: MONOALPHABETIC
SUBSTITUTION CIPHER");
        MonoalphabeticSubstitution_dec();
        break;

    case 4:
        f3=1;
        break;

    default:
        f3=0;
        if(indent==1)System.out.print("\t\t");
        System.out.println("Invalid Choice");
    }
}while(f3==0);

break;

case 7:
    f2=0;

    ch3=0;
    f3=0;
    do
    {
        temp=0;
        while(temp==0)
        {
            if(indent==1)System.out.print("\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t");
            System.out.println("HOME.2.7. POLYALPHABETIC SUBSTITUTION
CIPHER");

            if(indent==1)System.out.print("\t\t");
            System.out.println("1. About");
            if(indent==1)System.out.print("\t\t");
            System.out.println("2. Encrypt");
            if(indent==1)System.out.print("\t\t");
            System.out.println("3. Decrypt");
            if(indent==1)System.out.print("\t\t");
            System.out.println("4. Back To CIPHERS");
            if(indent==1)System.out.print("\t\t");
            System.out.print("Enter Choice: ");

```

```

        try
        {
            ch3=Integer.parseInt(bfrd.readLine());
            temp=1;
        }
        catch(NumberFormatException E1nfe)
        {
            if(indent==1)System.out.print("\t\t");
            System.out.println("Choice Should Be A Number");
            temp=0;
        }
    }

    switch(ch3)
    {
        case 1:
            f3=0;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("HOME.2.7.1. ABOUT: POLYALPHABETIC
SUBSTITUTION CIPHER");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("The Polyalphabetic Substitution Cipher is implemented");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("as a substitution cipher where each character is");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("replaced by another character which is obtained by");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("shifting the Plain Text character by the");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("corresponding Keyword character.");
            break;

        case 2:
            f3=0;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("HOME.2.7.2. ENCRYPTION: POLYALPHABETIC
SUBSTITUTION CIPHER");
            PolyalphabeticSubstitution_enc();
            break;

        case 3:
            f3=0;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("HOME.2.7.3. DECRYPTION: POLYALPHABETIC
SUBSTITUTION CIPHER");
            PolyalphabeticSubstitution_dec();
            break;
    }

```

```
        case 4:
            f3=1;
            break;

        default:
            f3=0;
            if(indent==1)System.out.print("\t\t");
            System.out.println("Invalid Choice");
        }
    }while(f3==0);

    break;

case 8:
    f2=0;

    ch3=0;
    f3=0;
    do
    {
        temp=0;
        while(temp==0)
        {
            if(indent==1)System.out.print("\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t");
            System.out.println("HOME.2.8. AUTOKEY CIPHER");
            if(indent==1)System.out.print("\t\t");
            System.out.println("1. About");
            if(indent==1)System.out.print("\t\t");
            System.out.println("2. Encrypt");
            if(indent==1)System.out.print("\t\t");
            System.out.println("3. Decrypt");
            if(indent==1)System.out.print("\t\t");
            System.out.println("4. Back To CIPHERS");
            if(indent==1)System.out.print("\t\t");
            System.out.print("Enter Choice: ");

            try
            {
                ch3=Integer.parseInt(bfrd.readLine());
                temp=1;
            }
            catch(NumberFormatException E1nfe)
            {
                if(indent==1)System.out.print("\t\t");
                System.out.println("Choice Should Be A Number");
                temp=0;
            }
        }
    }

    switch(ch3)
    {
        case 1:
```

```
f3=0;
if(indent==1)System.out.print("\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t");
System.out.println("HOME.2.8.1. ABOUT: AUTOKEY CIPHER");
if(indent==1)System.out.print("\t\t");
System.out.println("");
if(indent==1)System.out.print("\t\t");
System.out.println("The Autokey Substitution Cipher is");
if(indent==1)System.out.print("\t\t");
System.out.println("implemented as a substitution cipher");
if(indent==1)System.out.print("\t\t");
System.out.println("where each character is replaced by");
if(indent==1)System.out.print("\t\t");
System.out.println("another character which is obtained");
if(indent==1)System.out.print("\t\t");
System.out.println("by shifting the Plain Text character");
if(indent==1)System.out.print("\t\t");
System.out.println("by the corresponding 'Key' character.");
if(indent==1)System.out.print("\t\t");
System.out.println("The 'Key' is obtained from the 'Keyword'");
if(indent==1)System.out.print("\t\t");
System.out.println("by appending the Plain Text to the end");
if(indent==1)System.out.print("\t\t");
System.out.println("of the Keyword.");
break;

case 2:
    f3=0;
    if(indent==1)System.out.print("\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t");
    System.out.println("HOME.2.8.2. ENCRYPTION: AUTOKEY CIPHER");
    Autokey_enc();
    break;

case 3:
    f3=0;
    if(indent==1)System.out.print("\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t");
    System.out.println("HOME.2.8.3. DECRYPTION: AUTOKEY CIPHER");
    Autokey_dec();
    break;

case 4:
    f3=1;
    break;

default:
    f3=0;
    if(indent==1)System.out.print("\t\t");
    System.out.println("Invalid Choice");
}
}while(f3==0);
```

```
        break;

    case 9:
        f2=0;

        ch3=0;
        f3=0;
        do
        {
            temp=0;
            while(temp==0)
            {
                if(indent==1)System.out.print("\t\t");
                System.out.println();
                if(indent==1)System.out.print("\t\t");
                System.out.println("HOME.2.9. RUNNING KEY CIPHER");
                if(indent==1)System.out.print("\t\t");
                System.out.println("1. About");
                if(indent==1)System.out.print("\t\t");
                System.out.println("2. Encrypt");
                if(indent==1)System.out.print("\t\t");
                System.out.println("3. Decrypt");
                if(indent==1)System.out.print("\t\t");
                System.out.println("4. Back To CIPHERS");
                if(indent==1)System.out.print("\t\t");
                System.out.print("Enter Choice: ");

                try
                {
                    ch3=Integer.parseInt(bfrd.readLine());
                    temp=1;
                }
                catch(NumberFormatException E1nfe)
                {
                    if(indent==1)System.out.print("\t\t");
                    System.out.println("Choice Should Be A Number");
                    temp=0;
                }
            }
        }

        switch(ch3)
        {
            case 1:
                f3=0;
                if(indent==1)System.out.print("\t\t\t");
                System.out.println();
                if(indent==1)System.out.print("\t\t\t");
                System.out.println("HOME.2.9.1. ABOUT: RUNNING KEY CIPHER");
                if(indent==1)System.out.print("\t\t\t");
                System.out.println("");
                if(indent==1)System.out.print("\t\t\t");
                System.out.println("The Running Key Cipher is implemented");
                if(indent==1)System.out.print("\t\t\t");
                System.out.println("as a substitution cipher where each");
```

```
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("character is replaced by another");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("character which is obtained by shifting");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("the Plain Text character by the");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("corresponding Keyword character. For");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("this the length of the 'Keyword' should");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("be at least as long as the Plain Text.");
        break;

    case 2:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.9.2. ENCRYPTION: RUNNING KEY CIPHER");
        RunningKey_enc();
        break;

    case 3:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.9.3. DECRYPTION: RUNNING KEY CIPHER");
        RunningKey_dec();
        break;

    case 4:
        f3=1;
        break;

    default:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Choice");
    }
}while(f3==0);

break;

case 10:
    f2=0;

    ch3=0;
    f3=0;
    do
    {
        temp=0;
        while(temp==0)
        {
```

```

        if(indent==1)System.out.print("\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t");
        System.out.println("HOME.2.10. POLYBIUS SQUARE CIPHER");
        if(indent==1)System.out.print("\t\t");
        System.out.println("1. About");
        if(indent==1)System.out.print("\t\t");
        System.out.println("2. Encrypt");
        if(indent==1)System.out.print("\t\t");
        System.out.println("3. Decrypt");
        if(indent==1)System.out.print("\t\t");
        System.out.println("4. Back To CIPHERS");
        if(indent==1)System.out.print("\t\t");
        System.out.print("Enter Choice: ");

        try
        {
            ch3=Integer.parseInt(bfrd.readLine());
            temp=1;
        }
        catch(NumberFormatException E1nfe)
        {
            if(indent==1)System.out.print("\t\t");
            System.out.println("Choice Should Be A Number");
            temp=0;
        }
    }

    switch(ch3)
    {
        case 1:
            f3=0;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("HOME.2.10.1. ABOUT: POLYBIUS SQUARE CIPHER");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("The Polybius Square Cipher makes use of the");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Polybius square. The square has 10 rows and");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("10 columns each numbered from 0 through 9.");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Each element in the square is a character.");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("It is used to substitute the characters for");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("numbers corresponding to their location in");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("the square in the format 'row number', 'column");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("number'. The Cipher Text generated by the");
            if(indent==1)System.out.print("\t\t\t");

```



```

        System.out.println("program is a sequence of 3 digit ASCII values");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("of the Cipher Text characters.");
        break;

    case 2:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.10.2. ENCRYPTION: POLYBIUS SQUARE
CIPHER");

        PolybiusSquare_enc();
        break;

    case 3:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.10.3. DECRYPTION: POLYBIUS SQUARE
CIPHER");

        PolybiusSquare_dec();
        break;

    case 4:
        f3=1;
        break;

    default:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Choice");
    }
}while(f3==0);

break;

case 11:
    f2=0;

    ch3=0;
    f3=0;
    do
    {
        temp=0;
        while(temp==0)
        {
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("HOME.2.11. BIFID CIPHER");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("1. About");
            if(indent==1)System.out.print("\t\t\t");

```

```
System.out.println("2. Encrypt");
if(indent==1)System.out.print("\t\t");
System.out.println("3. Decrypt");
if(indent==1)System.out.print("\t\t");
System.out.println("4. Back To CIPHERS");
if(indent==1)System.out.print("\t\t");
System.out.print("Enter Choice: ");

try
{
    ch3=Integer.parseInt(bfrd.readLine());
    temp=1;
}
catch(NumberFormatException E1nfe)
{
    if(indent==1)System.out.print("\t\t");
    System.out.println("Choice Should Be A Number");
    temp=0;
}
}

switch(ch3)
{
    case 1:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.11.1. ABOUT: BIFID CIPHER");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("The Bifid Cipher makes use of the");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Polybius square. The square has 10");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("rows and 10 columns each numbered");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("from 0 through 9. Each element in");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("the square is a character. It is");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("used to substitute the characters");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("for numbers corresponding to their");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("location in the square while writing");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("all the row numbers first followed by");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("all the column numbers. The Cipher");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Text generated by the program is a");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("sequence of 3 digit ASCII values of");
```

```
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("the Cipher Text characters.");
        break;

    case 2:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.11.2. ENCRYPTION: BIFID CIPHER");
        Bifid_enc();
        break;

    case 3:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.11.3. DECRYPTION: BIFID CIPHER");
        Bifid_dec();
        break;

    case 4:
        f3=1;
        break;

    default:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Choice");
    }
}while(f3==0);

break;

case 12:
    f2=0;

    ch3=0;
    f3=0;
    do
    {
        temp=0;
        while(temp==0)
        {
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("HOME.2.12. TRIFID CIPHER");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("1. About");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("2. Encrypt");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("3. Decrypt");
```

```
if(indent==1)System.out.print("\t\t");
System.out.println("4. Back To CIPHERS");
if(indent==1)System.out.print("\t\t");
System.out.print("Enter Choice: ");

try
{
    ch3=Integer.parseInt(bfrd.readLine());
    temp=1;
}
catch(NumberFormatException E1nfe)
{
    if(indent==1)System.out.print("\t\t");
    System.out.println("Choice Should Be A Number");
    temp=0;
}
}

switch(ch3)
{
    case 1:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.12.1. ABOUT: TRIFID CIPHER");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("The Trifid Cipher uses a cuboid of");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("size 3 X 6 X 6. It can be viewed");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("as 3 squares (numbered from 0");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("through 2) with 6 rows (numbered from");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("0 through 5) and 6 columns (numbered");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("from 0 through 5). Each element in");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("the cuboid is a character. It is used");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("to substitute the characters for");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("numbers corresponding to their location");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("in the cuboid by writing all the square");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("numbers first, all the row numbers second");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("followed by all the column numbers third.");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("The Cipher Text generated by the program");
        if(indent==1)System.out.print("\t\t\t");
```

```
        System.out.println("is a sequence of 3 digit ASCII values of");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("the Cipher Text characters.");
        break;

    case 2:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.12.2. ENCRYPTION: TRIFID CIPHER");
        Trifid_enc();
        break;

    case 3:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.12.3. DECRYPTION: TRIFID CIPHER");
        Trifid_dec();
        break;

    case 4:
        f3=1;
        break;

    default:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Choice");
    }
}while(f3==0);

break;

case 13:
    f2=0;

    ch3=0;
    f3=0;
    do
    {
        temp=0;
        while(temp==0)
        {
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("HOME.2.13. BITWISE ROTATION CIPHER");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("1. About");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("2. Encrypt");
            if(indent==1)System.out.print("\t\t\t");
```

```

        System.out.println("3. Decrypt");
        if(indent==1)System.out.print("\t\t");
        System.out.println("4. Back To CIPHERS");
        if(indent==1)System.out.print("\t\t");
        System.out.print("Enter Choice: ");

        try
        {
            ch3=Integer.parseInt(bfrd.readLine());
            temp=1;
        }
        catch(NumberFormatException E1nfe)
        {
            if(indent==1)System.out.print("\t\t");
            System.out.println("Choice Should Be A Number");
            temp=0;
        }
    }

    switch(ch3)
    {
        case 1:
            f3=0;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("HOME.2.13.1. ABOUT: BITWISE ROTATION CIPHER");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("In the Bitwise Rotation Cipher the rotation");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("cipher is applied at the bit level. All the");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("ASCII characters can be represented by 8 bits");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("(1 byte). This gives us  $2^8 = 256$  possible");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("characters. The Cipher Text generated by the");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("program is a sequence of 3 digit ASCII values");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("of the Cipher Text characters.");
            break;

        case 2:
            f3=0;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("HOME.2.13.2. ENCRYPTION: BITWISE ROTATION");
            CIPHER");

            BitwiseRotation_enc();
            break;
    }

```

```

        case 3:
            f3=0;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("HOME.2.13.3. DECRYPTION: BITWISE ROTATION
CIPHER");

            BitwiseRotation_dec();
            break;

        case 4:
            f3=1;
            break;

        default:
            f3=0;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Invalid Choice");
        }
    }while(f3==0);

    break;

case 14:
    f2=0;

    ch3=0;
    f3=0;
    do
    {
        temp=0;
        while(temp==0)
        {
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("HOME.2.14. BITWISE ATBASH CIPHER");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("1. About");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("2. Encrypt");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("3. Decrypt");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("4. Back To CIPHERS");
            if(indent==1)System.out.print("\t\t\t");
            System.out.print("Enter Choice: ");

            try
            {
                ch3=Integer.parseInt(bfrd.readLine());
                temp=1;
            }
            catch(NumberFormatException E1nfe)
            {

```

```

        if(indent==1)System.out.print("\t\t");
        System.out.println("Choice Should Be A Number");
        temp=0;
    }
}

switch(ch3)
{
    case 1:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.14.1. ABOUT: BITWISE ATBASH CIPHER");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("In the Bitwise Atbash Cipher the atbash");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("cipher is applied at the bit level. All");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("the ASCII characters can be represented");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("by 8 bits (1 byte). This gives us 2^8 =");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("256 possible characters. The Cipher Text");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("generated by the program is a sequence of");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("3 digit ASCII values of the Cipher Text");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("characters.");
        break;

    case 2:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.14.2. ENCRYPTION: BITWISE ATBASH
CIPHER");

        BitwiseAtbash_enc();
        break;

    case 3:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.14.3. DECRYPTION: BITWISE ATBASH
CIPHER");

        BitwiseAtbash_dec();
        break;

    case 4:

```



```
        f3=1;
        break;

    default:
        f3=0;
        if(indent==1)System.out.print("\t\t");
        System.out.println("Invalid Choice");
    }
}while(f3==0);

break;

case 15:
    f2=0;

    ch3=0;
    f3=0;
    do
    {
        temp=0;
        while(temp==0)
        {
            if(indent==1)System.out.print("\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t");
            System.out.println("HOME.2.15. BITWISE BINARY COMPLIMENT CIPHER");
            if(indent==1)System.out.print("\t\t");
            System.out.println("1. About");
            if(indent==1)System.out.print("\t\t");
            System.out.println("2. Encrypt");
            if(indent==1)System.out.print("\t\t");
            System.out.println("3. Decrypt");
            if(indent==1)System.out.print("\t\t");
            System.out.println("4. Back To CIPHERS");
            if(indent==1)System.out.print("\t\t");
            System.out.print("Enter Choice: ");

            try
            {
                ch3=Integer.parseInt(bfrd.readLine());
                temp=1;
            }
            catch(NumberFormatException E1nfe)
            {
                if(indent==1)System.out.print("\t\t");
                System.out.println("Choice Should Be A Number");
                temp=0;
            }
        }
    }

    switch(ch3)
    {
        case 1:
            f3=0;
            if(indent==1)System.out.print("\t\t\t\t");
```

```

        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.15.1. ABOUT: BITWISE BINARY
COMPLIMENT CIPHER");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("In the Bitwise Binary Compliment Cipher each bit is");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("complimented, i.e., 1 becomes 0 and 0 becomes 1. All");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("the ASCII characters can be represented by 8 bits (1");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("byte). This gives us  $2^8 = 256$  possible characters. The");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Cipher Text generated by the program is a sequence of 3");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("digit ASCII values of the Cipher Text characters.");
        break;

    case 2:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.15.2. ENCRYPTION: BITWISE BINARY
COMPLIMENT CIPHER");
        BitwiseBinaryCompliment_enc();
        break;

    case 3:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.15.3. DECRYPTION: BITWISE BINARY
COMPLIMENT CIPHER");
        BitwiseBinaryCompliment_dec();
        break;

    case 4:
        f3=1;
        break;

    default:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Choice");
    }
}while(f3==0);

break;

case 16:
    f2=0;

```

```

ch3=0;
f3=0;
do
{
    temp=0;
    while(temp==0)
    {
        if(indent==1)System.out.print("\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t");
        System.out.println("HOME.2.16. BITWISE LEFT SHIFT CIPHER");
        if(indent==1)System.out.print("\t\t");
        System.out.println("1. About");
        if(indent==1)System.out.print("\t\t");
        System.out.println("2. Encrypt");
        if(indent==1)System.out.print("\t\t");
        System.out.println("3. Decrypt");
        if(indent==1)System.out.print("\t\t");
        System.out.println("4. Back To CIPHERS");
        if(indent==1)System.out.print("\t\t");
        System.out.print("Enter Choice: ");

        try
        {
            ch3=Integer.parseInt(bfrd.readLine());
            temp=1;
        }
        catch(NumberFormatException E1nfe)
        {
            if(indent==1)System.out.print("\t\t");
            System.out.println("Choice Should Be A Number");
            temp=0;
        }
    }

    switch(ch3)
    {
        case 1:
            f3=0;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("HOME.2.16.1. ABOUT: BITWISE LEFT SHIFT
CIPHER");

            if(indent==1)System.out.print("\t\t\t");
            System.out.println("");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("In the Bitwise Left Shift Cipher each bit is");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("shifted to the left by the number in the Keyword");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("(0-7). All the ASCII characters can be represented");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("by 8 bits (1 byte). This gives us 2^8 = 256 possible");

```

```

        if(indent==1)System.out.print("\t\t\t");
        System.out.println("characters. The Cipher Text generated by the program");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("is a sequence of 3 digit ASCII values of the Cipher");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Text characters.");
        break;

    case 2:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.16.2. ENCRYPTION: BITWISE LEFT SHIFT
CIPHER");

        BitwiseLeftShift_enc();
        break;

    case 3:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.16.3. DECRYPTION: BITWISE LEFT SHIFT
CIPHER");

        BitwiseLeftShift_dec();
        break;

    case 4:
        f3=1;
        break;

    default:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Choice");
    }
}while(f3==0);

break;

case 17:
    f2=0;

    ch3=0;
    f3=0;
    do
    {
        temp=0;
        while(temp==0)
        {
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("HOME.2.17. BITWISE RIGHT SHIFT CIPHER");

```

```

        if(indent==1)System.out.print("\t\t");
        System.out.println("1. About");
        if(indent==1)System.out.print("\t\t");
        System.out.println("2. Encrypt");
        if(indent==1)System.out.print("\t\t");
        System.out.println("3. Decrypt");
        if(indent==1)System.out.print("\t\t");
        System.out.println("4. Back To CIPHERS");
        if(indent==1)System.out.print("\t\t");
        System.out.print("Enter Choice: ");

        try
        {
            ch3=Integer.parseInt(bfrd.readLine());
            temp=1;
        }
        catch(NumberFormatException E1nfe)
        {
            if(indent==1)System.out.print("\t\t");
            System.out.println("Choice Should Be A Number");
            temp=0;
        }
    }

    switch(ch3)
    {
        case 1:
            f3=0;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("HOME.2.17.1. ABOUT: BITWISE RIGHT SHIFT
CIPHER");

            if(indent==1)System.out.print("\t\t\t");
            System.out.println("");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("In the Bitwise Right Shift Cipher each bit is");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("shifted to the right by the number in the Keyword");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("(0-7). All the ASCII characters can be represented");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("by 8 bits (1 byte). This gives us  $2^8 = 256$  possible");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("characters. The Cipher Text generated by the program");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("is a sequence of 3 digit ASCII values of the Cipher");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Text characters.");
            break;

        case 2:
            f3=0;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();

```

```

        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.17.2. ENCRYPTION: BITWISE RIGHT SHIFT
CIPHER");

        BitwiseRightShift_enc();
        break;

    case 3:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.17.3. DECRYPTION: BITWISE RIGHT SHIFT
CIPHER");

        BitwiseRightShift_dec();
        break;

    case 4:
        f3=1;
        break;

    default:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Choice");
    }
}while(f3==0);

break;

case 18:
    f2=0;

    ch3=0;
    f3=0;
    do
    {
        temp=0;
        while(temp==0)
        {
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("HOME.2.18. BITWISE EXCLUSIVE OR CIPHER");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("1. About");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("2. Encrypt");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("3. Decrypt");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("4. Back To CIPHERS");
            if(indent==1)System.out.print("\t\t\t");
            System.out.print("Enter Choice: ");

            try

```

```

        {
            ch3=Integer.parseInt(bfrd.readLine());
            temp=1;
        }
        catch(NumberFormatException E1nfe)
        {
            if(indent==1)System.out.print("\t\t");
            System.out.println("Choice Should Be A Number");
            temp=0;
        }
    }

    switch(ch3)
    {
        case 1:
            f3=0;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("HOME.2.18.1. ABOUT: BITWISE EXCLUSIVE OR
CIPHER");

            if(indent==1)System.out.print("\t\t\t");
            System.out.println("");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("In the Bitwise Exclusive Or Cipher the logical");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("XOR (Exclusive Or) operator is used. The");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("corresponding bits of the Keyword and the Plain");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Text on applying the XOR operation give the Cipher");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Text. All the ASCII characters can be represented");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("by 8 bits (1 byte). This gives us 2^8 = 256 possible");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("characters. The Cipher Text generated by the program");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("is a sequence of 3 digit ASCII values of the Cipher");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Text characters.");
            break;

        case 2:
            f3=0;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("HOME.2.18.2. ENCRYPTION: BITWISE EXCLUSIVE OR
CIPHER");

            BitwiseExclusiveOr_enc();
            break;

        case 3:
            f3=0;

```

```

        if(indent==1)System.out.print("\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t");
        System.out.println("HOME.2.18.3. DECRYPTION: BITWISE EXCLUSIVE OR
CIPHER");

        BitwiseExclusiveOr_dec();
        break;

    case 4:
        f3=1;
        break;

    default:
        f3=0;
        if(indent==1)System.out.print("\t\t");
        System.out.println("Invalid Choice");
    }
}while(f3==0);

break;

case 19:
    f2=0;

    ch3=0;
    f3=0;
    do
    {
        temp=0;
        while(temp==0)
        {
            if(indent==1)System.out.print("\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t");
            System.out.println("HOME.2.19. BITWISE ONE TIME PAD CIPHER");
            if(indent==1)System.out.print("\t\t");
            System.out.println("1. About");
            if(indent==1)System.out.print("\t\t");
            System.out.println("2. Encrypt");
            if(indent==1)System.out.print("\t\t");
            System.out.println("3. Decrypt");
            if(indent==1)System.out.print("\t\t");
            System.out.println("4. Back To CIPHERS");
            if(indent==1)System.out.print("\t\t");
            System.out.print("Enter Choice: ");

            try
            {
                ch3=Integer.parseInt(bfrd.readLine());
                temp=1;
            }
            catch(NumberFormatException E1nfe)
            {
                if(indent==1)System.out.print("\t\t");
                System.out.println("Choice Should Be A Number");
            }
        }
    }

```



```

        temp=0;
    }
}

switch(ch3)
{
    case 1:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.19.1. ABOUT: BITWISE ONE TIME PAD
CIPHER");

        if(indent==1)System.out.print("\t\t\t");
        System.out.println("");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("In the Bitwise One Time Pad Cipher the logical");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("XOR (Exclusive Or) operator is used. The");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("corresponding bits of the Keyword and the Plain");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Text on applying the XOR operation give the Cipher");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Text. The Keyword must be at least as long as the");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Plain Text. All the ASCII characters can be");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("represented by 8 bits (1 byte). This gives us  $2^8 =$ ");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("256 possible characters. The Cipher Text generated");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("by the program is a sequence of 3 digit ASCII values");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("of the Cipher Text characters.");
        break;

    case 2:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.19.2. ENCRYPTION: BITWISE ONE TIME PAD
CIPHER");

        BitwiseOneTimePad_enc();
        break;

    case 3:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.19.3. DECRYPTION: BITWISE ONE TIME PAD
CIPHER");

        BitwiseOneTimePad_dec();

```

```
        break;

    case 4:
        f3=1;
        break;

    default:
        f3=0;
        if(indent==1)System.out.print("\t\t");
        System.out.println("Invalid Choice");
    }
}while(f3==0);

break;

case 20:
    f2=0;

    ch3=0;
    f3=0;
    do
    {
        temp=0;
        while(temp==0)
        {
            if(indent==1)System.out.print("\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t");
            System.out.println("HOME.2.20. SQUARE TRANSPOSITION CIPHER");
            if(indent==1)System.out.print("\t\t");
            System.out.println("1. About");
            if(indent==1)System.out.print("\t\t");
            System.out.println("2. Encrypt");
            if(indent==1)System.out.print("\t\t");
            System.out.println("3. Decrypt");
            if(indent==1)System.out.print("\t\t");
            System.out.println("4. Back To CIPHERS");
            if(indent==1)System.out.print("\t\t");
            System.out.print("Enter Choice: ");

            try
            {
                ch3=Integer.parseInt(bfrd.readLine());
                temp=1;
            }
            catch(NumberFormatException E1nfe)
            {
                if(indent==1)System.out.print("\t\t");
                System.out.println("Choice Should Be A Number");
                temp=0;
            }
        }
    }

    switch(ch3)
    {
```

```
case 1:
    f3=0;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("HOME.2.20.1. ABOUT: SQUARE TRANSPOSITION
CIPHER");

    if(indent==1)System.out.print("\t\t\t");
    System.out.println("");
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("The Square Transposition cipher arranges the");
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Plain Text in a perfect square, i.e., number of");
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("rows equal the number of columns, where each");
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("element in the square has a character. If there");
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("are empty elements they are padded with spaces.");
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("The cipher is obtained by interchanging the rows");
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("with the columns.");
    break;

case 2:
    f3=0;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("HOME.2.20.2. ENCRYPTION: SQUARE
TRANSPOSITION CIPHER");
    SquareTransposition_enc();
    break;

case 3:
    f3=0;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("HOME.2.20.3. DECRYPTION: SQUARE
TRANSPOSITION CIPHER");
    SquareTransposition_dec();
    break;

case 4:
    f3=1;
    break;

default:
    f3=0;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Invalid Choice");
}
}while(f3==0);
```

```

        break;

    case 21:
        f2=0;

        ch3=0;
        f3=0;
        do
        {
            temp=0;
            while(temp==0)
            {
                if(indent==1)System.out.print("\t\t");
                System.out.println();
                if(indent==1)System.out.print("\t\t");
                System.out.println("HOME.2.21. COLUMNAR TRANSPOSITION CIPHER");
                if(indent==1)System.out.print("\t\t");
                System.out.println("1. About");
                if(indent==1)System.out.print("\t\t");
                System.out.println("2. Encrypt");
                if(indent==1)System.out.print("\t\t");
                System.out.println("3. Decrypt");
                if(indent==1)System.out.print("\t\t");
                System.out.println("4. Back To CIPHERS");
                if(indent==1)System.out.print("\t\t");
                System.out.print("Enter Choice: ");

                try
                {
                    ch3=Integer.parseInt(bfrd.readLine());
                    temp=1;
                }
                catch(NumberFormatException E1nfe)
                {
                    if(indent==1)System.out.print("\t\t");
                    System.out.println("Choice Should Be A Number");
                    temp=0;
                }
            }
        }

        switch(ch3)
        {
            case 1:
                f3=0;
                if(indent==1)System.out.print("\t\t\t");
                System.out.println();
                if(indent==1)System.out.print("\t\t\t");
                System.out.println("HOME.2.21.1. ABOUT: COLUMNAR TRANSPOSITION
CIPHER");

                if(indent==1)System.out.print("\t\t\t");
                System.out.println("");
                if(indent==1)System.out.print("\t\t\t");
                System.out.println("In a columnar transposition, the message is written");
                if(indent==1)System.out.print("\t\t\t");

```

```

        System.out.println("out in rows of a fixed length, and then read out");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("again column by column, and the columns are chosen");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("in some scrambled order. Both the width of the rows");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("and the permutation of the columns are based on the");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Keyword.");
        break;

    case 2:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.21.2. ENCRYPTION: COLUMNAR
TRANSPPOSITION CIPHER");
        ColumnarTransposition_enc();
        break;

    case 3:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.21.3. DECRYPTION: COLUMNAR
TRANSPPOSITION CIPHER");
        ColumnarTransposition_dec();
        break;

    case 4:
        f3=1;
        break;

    default:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Choice");
    }
    }while(f3==0);

    break;

case 22:
    f2=0;

    ch3=0;
    f3=0;
    do
    {
        temp=0;
        while(temp==0)
        {
            if(indent==1)System.out.print("\t\t\t");

```

```

        System.out.println();
        if(indent==1)System.out.print("\t\t");
        System.out.println("HOME.2.22. DOUBLE TRANSPOSITION CIPHER");
        if(indent==1)System.out.print("\t\t");
        System.out.println("1. About");
        if(indent==1)System.out.print("\t\t");
        System.out.println("2. Encrypt");
        if(indent==1)System.out.print("\t\t");
        System.out.println("3. Decrypt");
        if(indent==1)System.out.print("\t\t");
        System.out.println("4. Back To CIPHERS");
        if(indent==1)System.out.print("\t\t");
        System.out.print("Enter Choice: ");

        try
        {
            ch3=Integer.parseInt(bfrd.readLine());
            temp=1;
        }
        catch(NumberFormatException E1nfe)
        {
            if(indent==1)System.out.print("\t\t");
            System.out.println("Choice Should Be A Number");
            temp=0;
        }
    }

    switch(ch3)
    {
        case 1:
            f3=0;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("HOME.2.22.1. ABOUT: DOUBLE TRANSPOSITION
CIPHER");

            if(indent==1)System.out.print("\t\t\t");
            System.out.println("");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("The Double Transposition Cipher is simply the");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Columnar Transposition Cipher applied twice.");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("The same Keyword can be used for both");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("transpositions or two different Keywords can");
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("be used.");
            break;

        case 2:
            f3=0;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t\t");

```

```
        System.out.println("HOME.2.22.2. ENCRYPTION: DOUBLE  
TRANSPOSITION CIPHER");  
        DoubleTransposition_enc();  
        break;  
  
    case 3:  
        f3=0;  
        if(indent==1)System.out.print("\t\t");  
        System.out.println();  
        if(indent==1)System.out.print("\t\t");  
        System.out.println("HOME.2.22.3. DECRYPTION: DOUBLE  
TRANSPOSITION CIPHER");  
        DoubleTransposition_dec();  
        break;  
  
    case 4:  
        f3=1;  
        break;  
  
    default:  
        f3=0;  
        if(indent==1)System.out.print("\t\t");  
        System.out.println("Invalid Choice");  
    }  
    }while(f3==0);  
  
    break;  
  
case 23:  
    f2=0;  
  
    ch3=0;  
    f3=0;  
    do  
    {  
        temp=0;  
        while(temp==0)  
        {  
            if(indent==1)System.out.print("\t\t");  
            System.out.println();  
            if(indent==1)System.out.print("\t\t");  
            System.out.println("HOME.2.23. MYSZKOWSKI TRANSPOSITION CIPHER");  
            if(indent==1)System.out.print("\t\t");  
            System.out.println("1. About");  
            if(indent==1)System.out.print("\t\t");  
            System.out.println("2. Encrypt");  
            if(indent==1)System.out.print("\t\t");  
            System.out.println("3. Decrypt");  
            if(indent==1)System.out.print("\t\t");  
            System.out.println("4. Back To CIPHERS");  
            if(indent==1)System.out.print("\t\t");  
            System.out.print("Enter Choice: ");  
  
            try  
            {
```

```

        ch3=Integer.parseInt(bfrd.readLine());
        temp=1;
    }
    catch(NumberFormatException E1nfe)
    {
        if(indent==1)System.out.print("\t\t");
        System.out.println("Choice Should Be A Number");
        temp=0;
    }
}

switch(ch3)
{
    case 1:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.23.1. ABOUT: MYSZKOWSKI
TRANSPPOSITION CIPHER");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("The Myszowski Transposition is a variant form of");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Columnar Transposition, proposed by Émile Victor");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Théodore Myszowski in 1902. It requires a keyword");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("with recurrent letters. In Columnar Transposition");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("subsequent occurrences of a keyword letter are");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("treated as if the next letter in alphabetical");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("order, e.g., the keyword TOMATO yields a numeric");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("keysting of '532164'. In Myszowski Transposition,");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("recurrent keyword letters are numbered identically,");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("thus TOMATO yielding a keysting of '432143.'");
        break;

    case 2:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.23.2. ENCRYPTION: MYSZKOWSKI
TRANSPPOSITION CIPHER");
        MyszowskiTransposition_enc();
        break;

    case 3:

```



```

        f3=0;
        if(indent==1)System.out.print("\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t");
        System.out.println("HOME.2.23.3. DECRYPTION: MYSZKOWSKI
TRANSPOSITION CIPHER");
        MyszowskiTransposition_dec();
        break;

    case 4:
        f3=1;
        break;

    default:
        f3=0;
        if(indent==1)System.out.print("\t\t");
        System.out.println("Invalid Choice");
    }
}while(f3==0);

break;

case 24:
    f2=0;

    ch3=0;
    f3=0;
    do
    {
        temp=0;
        while(temp==0)
        {
            if(indent==1)System.out.print("\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t");
            System.out.println("HOME.2.24. CHINESE TRANSPOSITION CIPHER");
            if(indent==1)System.out.print("\t\t");
            System.out.println("1. About");
            if(indent==1)System.out.print("\t\t");
            System.out.println("2. Encrypt");
            if(indent==1)System.out.print("\t\t");
            System.out.println("3. Decrypt");
            if(indent==1)System.out.print("\t\t");
            System.out.println("4. Back To CIPHERS");
            if(indent==1)System.out.print("\t\t");
            System.out.print("Enter Choice: ");

            try
            {
                ch3=Integer.parseInt(bfrd.readLine());
                temp=1;
            }
            catch(NumberFormatException E1nfe)
            {
                if(indent==1)System.out.print("\t\t");

```

```

        System.out.println("Choice Should Be A Number");
        temp=0;
    }
}

switch(ch3)
{
    case 1:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.24.1. ABOUT: CHINESE TRANSPOSITION
CIPHER");

        if(indent==1)System.out.print("\t\t\t");
        System.out.println("");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("In the Chinese Transposition Cipher, the letters");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("of the message are written from right to left,");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("down and up columns to scramble the letters.");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Then, starting in the first row, the letters");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("are taken in order to get the new Cipher Text.");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("The Keyword determined the number of rows. All");
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("remaining blanks are filled with spaces.");
        break;

    case 2:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.24.2. ENCRYPTION: CHINESE
TRANSPOSITION CIPHER");
        ChineseTransposition_enc();
        break;

    case 3:
        f3=0;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("HOME.2.24.3. DECRYPTION: CHINESE
TRANSPOSITION CIPHER");
        ChineseTransposition_dec();
        break;

    case 4:
        f3=1;
        break;

```

```
        default:
            f3=0;
            if(indent==1)System.out.print("\t\t");
            System.out.println("Invalid Choice");
        }
    }while(f3==0);

    break;

case 25:
    f2=0;

    ch3=0;
    f3=0;
    do
    {
        temp=0;
        while(temp==0)
        {
            if(indent==1)System.out.print("\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t");
            System.out.println("HOME.2.25. RAIL FENCE CIPHER");
            if(indent==1)System.out.print("\t\t");
            System.out.println("1. About");
            if(indent==1)System.out.print("\t\t");
            System.out.println("2. Encrypt");
            if(indent==1)System.out.print("\t\t");
            System.out.println("3. Decrypt");
            if(indent==1)System.out.print("\t\t");
            System.out.println("4. Back To CIPHERS");
            if(indent==1)System.out.print("\t\t");
            System.out.print("Enter Choice: ");

            try
            {
                ch3=Integer.parseInt(bfrd.readLine());
                temp=1;
            }
            catch(NumberFormatException E1nfe)
            {
                if(indent==1)System.out.print("\t\t");
                System.out.println("Choice Should Be A Number");
                temp=0;
            }
        }
    }

    switch(ch3)
    {
        case 1:
            f3=0;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t\t");
```

```
System.out.println("HOME.2.25.1. ABOUT: RAIL FENCE CIPHER");
if(indent==1)System.out.print("\t\t");
System.out.println("");
if(indent==1)System.out.print("\t\t");
System.out.println("The Rail Fence Cipher is a form of");
if(indent==1)System.out.print("\t\t");
System.out.println("transposition cipher that gets its");
if(indent==1)System.out.print("\t\t");
System.out.println("name from the way in which it is");
if(indent==1)System.out.print("\t\t");
System.out.println("encoded. In the rail fence cipher, the");
if(indent==1)System.out.print("\t\t");
System.out.println("Plain Text is written downwards on");
if(indent==1)System.out.print("\t\t");
System.out.println("successive 'rails' of an imaginary fence,");
if(indent==1)System.out.print("\t\t");
System.out.println("then moving up when we get to the");
if(indent==1)System.out.print("\t\t");
System.out.println("bottom. Blanks are filled with spaces.");
if(indent==1)System.out.print("\t\t");
System.out.println("The message is then read off in rows");
if(indent==1)System.out.print("\t\t");
System.out.println("from left to right.");
break;

case 2:
    f3=0;
    if(indent==1)System.out.print("\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t");
    System.out.println("HOME.2.25.2. ENCRYPTION: RAIL FENCE CIPHER");
    RailFence_enc();
    break;

case 3:
    f3=0;
    if(indent==1)System.out.print("\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t");
    System.out.println("HOME.2.25.3. DECRYPTION: RAIL FENCE CIPHER");
    RailFence_dec();
    break;

case 4:
    f3=1;
    break;

default:
    f3=0;
    if(indent==1)System.out.print("\t\t");
    System.out.println("Invalid Choice");
}
}while(f3==0);

break;
```

```
        case 26:
            f2=1;
            break;

        default:
            f2=0;
            if(indent==1)System.out.print("\t");
            System.out.println("Invalid Choice");
        }
    }while(f2==0);

    break;

case 3:
    f1=0;
    if(indent==1)System.out.print("\t");
    System.out.println();
    if(indent==1)System.out.print("\t");
    System.out.println("HOME.3. AUTHOR");
    if(indent==1)System.out.print("\t");
    System.out.println("Name: Omkar S. Nath");
    if(indent==1)System.out.print("\t");
    System.out.println("Class: 10–D");
    if(indent==1)System.out.print("\t");
    System.out.println("Roll Number: 35");
    if(indent==1)System.out.print("\t");
    System.out.println("Computer Code: 230438");
    break;

case 4:
    f1=1;
    if(indent==1)System.out.print("\t");
    System.out.println();
    if(indent==1)System.out.print("\t");
    System.out.println("Thank You For Using This Program.");
    if(indent==1)System.out.print("\t");
    System.out.println("- Omkar S. Nath");
    break;

default:
    f1=0;
    System.out.println("Invalid Choice");
}
}while(f1==0);
}

public static void Rotation_enc()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int ptl;
    int key;
    int temp;
```

```
int pti[];
pti = new int[10000];

int cti[];
cti = new int[10000];

char ptc[];
ptc = new char[10000];

char ctc[];
ctc = new char[10000];

String pt;

key=48;

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Plain Text (A String Of Length Greater Than 2):");
    if(indent==1)System.out.print("\t\t\t");
    pt=bfrd.readLine();
    ptl=pt.length();
    if(ptl<3)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Plain Text Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<ptl;i++)
{
    ptc[i]=pt.charAt(i);
    pti[i]=ptc[i];
}

for(i=0;i<ptl;i++)
{
    cti[i]=pti[i]-32;
    cti[i]=cti[i]+key;
    cti[i]=cti[i]%95;
    cti[i]=cti[i]+32;
}

if(indent==1)System.out.print("\t\t\t");
System.out.println();
```

```
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Cipher Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<ptl;i++)
{
    ctc[i]=(char)cti[i];
    System.out.print(ctc[i]);
}
System.out.println("");
}

public static void Rotation_dec()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int ctl;
    int key;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    char ptc[];
    ptc = new char[10000];

    char ctc[];
    ctc = new char[10000];

    String ct;

    key=48;

    temp=0;
    do
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter The Cipher Text (A String Of Length Greater Than 2):");
        if(indent==1)System.out.print("\t\t\t");
        ct=bfrd.readLine();
        ctl=ct.length();
        if(ctl<3)
        {
            temp=1;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Invalid Cipher Text Length");
        }
        else
        {
```

```
        temp=0;
    }
} while(temp==1);

for(i=0;i<ctl;i++)
{
    ctc[i]=ct.charAt(i);
    cti[i]=ctc[i];
}

for(i=0;i<ctl;i++)
{
    pti[i]=cti[i]-32;
    pti[i]=pti[i]-key%95;
    if(pti[i]<0)
    {
        pti[i]=pti[i]+95;
    }
    pti[i]=pti[i]%95;
    pti[i]=pti[i]+32;
}

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Plain Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<ctl;i++)
{
    ptc[i]=(char)pti[i];
    System.out.print(ptc[i]);
}
System.out.println("");
}

public static void Atbash_enc()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int ptl;
    int numofchar;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    char ptc[];
    ptc = new char[10000];

    char ctc[];
```



```
ctc = new char[10000];

String pt;

numofchar=94;

temp=0;
do
{
    if(indent==1)System.out.print("\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t");
    System.out.println("Enter The Plain Text (A String Of Length Greater Than 2):");
    if(indent==1)System.out.print("\t\t");
    pt=bfrd.readLine();
    ptl=pt.length();
    if(ptl<3)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t");
        System.out.println("Invalid Plain Text Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<ptl;i++)
{
    ptc[i]=pt.charAt(i);
    pti[i]=ptc[i];
}

for(i=0;i<ptl;i++)
{
    cti[i]=pti[i]-32;
    cti[i]=numofchar-cti[i];
    cti[i]=cti[i]+32;
}

if(indent==1)System.out.print("\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t");
System.out.println("The Cipher Text Is:");
if(indent==1)System.out.print("\t\t");
System.out.print("");
for(i=0;i<ptl;i++)
{
    ctc[i]=(char)cti[i];
    System.out.print(ctc[i]);
}
System.out.println("");
}
```

```
public static void Atbash_dec()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int ctl;
    int numofchar;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    char ptc[];
    ptc = new char[10000];

    char ctc[];
    ctc = new char[10000];

    String ct;

    numofchar=94;

    temp=0;
    do
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter The Cipher Text (A String Of Length Greater Than 2):");
        if(indent==1)System.out.print("\t\t\t");
        ct=bfrd.readLine();
        ctl=ct.length();
        if(ctl<3)
        {
            temp=1;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Invalid Cipher Text Length");
        }
        else
        {
            temp=0;
        }
    }while(temp==1);

    for(i=0;i<ctl;i++)
    {
        ctc[i]=ct.charAt(i);
        cti[i]=ctc[i];
    }

    for(i=0;i<ctl;i++)
    {
```

```
        pti[i]=cti[i]-32;
        pti[i]=numofchar-pti[i];
        pti[i]=pti[i]+32;
    }

    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("The Plain Text Is:");
    if(indent==1)System.out.print("\t\t\t");
    System.out.print("");
    for(i=0;i<ctl;i++)
    {
        ptc[i]=(char)pti[i];
        System.out.print(ptc[i]);
    }
    System.out.println("");
}

public static void CaesarPositiveShift_enc()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int ptl;
    int key;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    char ptc[];
    ptc = new char[10000];

    char ctc[];
    ctc = new char[10000];

    String pt;

    key=0;

    temp=0;
    do
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter The Plain Text (A String Of Length Greater Than 2):");
        if(indent==1)System.out.print("\t\t\t");
        pt=bfrd.readLine();
        ptl=pt.length();
        if(ptl<3)
```

```
{
    temp=1;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Invalid Plain Text Length");
}
else
{
    temp=0;
}
}while(temp==1);

do
{
    temp=0;
    while(temp==0)
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter the Key (A Number Between 0 and 10000):");
        if(indent==1)System.out.print("\t\t\t");
        try
        {
            key=Integer.parseInt(bfrd.readLine());
            temp=1;
        }
        catch(NumberFormatException E1nfe)
        {
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Key Should Be A Number");
            temp=0;
        }
    }
}

if(key<1||key>9999)
{
    temp=1;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Invalid Key");
}
else
{
    temp=0;
}
}while(temp==1);

for(i=0;i<ptl;i++)
{
    ptc[i]=pt.charAt(i);
    pti[i]=ptc[i];
}

for(i=0;i<ptl;i++)
{
    cti[i]=pti[i]-32;
```

```
        cti[i]=cti[i]+key;
        cti[i]=cti[i]%95;
        cti[i]=cti[i]+32;
    }

    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("The Cipher Text Is:");
    if(indent==1)System.out.print("\t\t\t");
    System.out.print("");
    for(i=0;i<ptl;i++)
    {
        ctc[i]=(char)cti[i];
        System.out.print(ctc[i]);
    }
    System.out.println("");
}

public static void CaesarPositiveShift_dec()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int ctl;
    int key;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    char ptc[];
    ptc = new char[10000];

    char ctc[];
    ctc = new char[10000];

    String ct;

    key=0;

    temp=0;
    do
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter The Cipher Text (A String Of Length Greater Than 2):");
        if(indent==1)System.out.print("\t\t\t");
        ct=bfrd.readLine();
        ctl=ct.length();
        if(ctl<3)
```

```
{
    temp=1;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Invalid Cipher Text Length");
}
else
{
    temp=0;
}
}while(temp==1);

do
{
    temp=0;
    while(temp==0)
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter the Key (A Number Between 0 and 10000):");
        if(indent==1)System.out.print("\t\t\t");
        try
        {
            key=Integer.parseInt(bfrd.readLine());
            temp=1;
        }
        catch(NumberFormatException E1nfe)
        {
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Key Should Be A Number");
            temp=0;
        }
    }
}

if(key<1||key>9999)
{
    temp=1;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Invalid Key");
}
else
{
    temp=0;
}
}while(temp==1);

for(i=0;i<ctl;i++)
{
    ctc[i]=ct.charAt(i);
    cti[i]=ctc[i];
}

for(i=0;i<ctl;i++)
{
    pti[i]=cti[i]-32;
```

```
        pti[i]=pti[i]-key%95;
        if(pti[i]<0)
        {
            pti[i]=pti[i]+95;
        }
        pti[i]=pti[i]%95;
        pti[i]=pti[i]+32;
    }

    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("The Plain Text Is:");
    if(indent==1)System.out.print("\t\t\t");
    System.out.print("");
    for(i=0;i<ctl;i++)
    {
        ptc[i]=(char)pti[i];
        System.out.print(ptc[i]);
    }
    System.out.println("");
}

public static void CaesarNegativeShift_enc()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int ptl;
    int key;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    char ptc[];
    ptc = new char[10000];

    char ctc[];
    ctc = new char[10000];

    String pt;

    key=0;

    temp=0;
    do
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter The Plain Text (A String Of Length Greater Than 2):");
```

```
if(indent==1)System.out.print("\t\t\t");
pt=bfrd.readLine();
ptl=pt.length();
if(ptl<3)
{
    temp=1;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Invalid Plain Text Length");
}
else
{
    temp=0;
}
}while(temp==1);

do
{
    temp=0;
    while(temp==0)
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter the Key (A Number Between 0 and 10000):");
        if(indent==1)System.out.print("\t\t\t");
        try
        {
            key=Integer.parseInt(bfrd.readLine());
            temp=1;
        }
        catch(NumberFormatException E1nfe)
        {
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Key Should Be A Number");
            temp=0;
        }
    }

    if(key<1||key>9999)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Key");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<ptl;i++)
{
    ptc[i]=pt.charAt(i);
    pti[i]=ptc[i];
}
```



```
for(i=0;i<ptl;i++)
{
    cti[i]=pti[i]-32;
    cti[i]=cti[i]-key%95;
    if(cti[i]<0)
    {
        cti[i]=cti[i]+95;
    }
    cti[i]=cti[i]%95;
    cti[i]=cti[i]+32;
}

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Cipher Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<ptl;i++)
{
    ctc[i]=(char)cti[i];
    System.out.print(ctc[i]);
}
System.out.println("");
}

public static void CaesarNegativeShift_dec()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int ctl;
    int key;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    char ptc[];
    ptc = new char[10000];

    char ctc[];
    ctc = new char[10000];

    String ct;

    key=0;

    temp=0;
    do
    {
```

```
if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("Enter The Cipher Text (A String Of Length Greater Than 2):");
if(indent==1)System.out.print("\t\t\t");
ct=bfrd.readLine();
ctl=ct.length();
if(ctl<3)
{
    temp=1;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Invalid Cipher Text Length");
}
else
{
    temp=0;
}
}while(temp==1);

temp=0;
do
{
    temp=0;
    while(temp==0)
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter the Key (A Number Between 0 and 10000):");
        if(indent==1)System.out.print("\t\t\t");
        try
        {
            key=Integer.parseInt(bfrd.readLine());
            temp=1;
        }
        catch(NumberFormatException E1nfe)
        {
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Key Should Be A Number");
            temp=0;
        }
    }
}

if(key<1||key>9999)
{
    temp=1;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Invalid Key");
}
else
{
    temp=0;
}
}while(temp==1);
```

```
for(i=0;i<ctl;i++)
{
    ctc[i]=ct.charAt(i);
    cti[i]=ctc[i];
}

for(i=0;i<ctl;i++)
{
    pti[i]=cti[i]-32;
    pti[i]=pti[i]+key;
    pti[i]=pti[i]%95;
    pti[i]=pti[i]+32;
}

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Plain Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<ctl;i++)
{
    ptc[i]=(char)pti[i];
    System.out.print(ptc[i]);
}
System.out.println("");
}

public static void OneTimePad_enc()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int ptl;
    int keyl;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int keyi[];
    keyi = new int[10000];

    char ptc[];
    ptc = new char[10000];

    char ctc[];
    ctc = new char[10000];

    char keyc[];
    keyc = new char[10000];
```

```
String pt;
String key;

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Plain Text (A String Of Length Greater Than 2):");
    if(indent==1)System.out.print("\t\t\t");
    pt=bfrd.readLine();
    ptl=pt.length();
    if(ptl<3)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Plain Text Length");
    }
    else
    {
        temp=0;
    }
} while(temp==1);

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Keyword (A String Of Length Atleast As Long As Plain Text):");
    if(indent==1)System.out.print("\t\t\t");
    key=bfrd.readLine();
    keyl=key.length();
    if(keyl<ptl)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Key Length");
    }
    else
    {
        temp=0;
    }
} while(temp==1);

for(i=0;i<ptl;i++)
{
    ptc[i]=pt.charAt(i);
    pti[i]=ptc[i];
}

for(i=0;i<keyl;i++)
{
```

```
        keyc[i]=key.charAt(i);
        keyi[i]=keyc[i];
        keyi[i]=keyi[i]-32;
    }

    for(i=0;i<ptl;i++)
    {
        cti[i]=pti[i]-32;
        cti[i]=cti[i]+keyi[i];
        cti[i]=cti[i]%95;
        cti[i]=cti[i]+32;
    }

    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("The Cipher Text Is:");
    if(indent==1)System.out.print("\t\t\t");
    System.out.print("");
    for(i=0;i<ptl;i++)
    {
        ctc[i]=(char)cti[i];
        System.out.print(ctc[i]);
    }
    System.out.println("");
}

public static void OneTimePad_dec()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int ctl;
    int keyl;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int keyi[];
    keyi = new int[10000];

    char ptc[];
    ptc = new char[10000];

    char ctc[];
    ctc = new char[10000];

    char keyc[];
    keyc = new char[10000];

    String ct;
```

```
String key;

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Cipher Text (A String Of Length Greater Than 2):");
    if(indent==1)System.out.print("\t\t\t");
    ct=bfrd.readLine();
    ctl=ct.length();
    if(ctl<3)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Cipher Text Length");
    }
    else
    {
        temp=0;
    }
} while(temp==1);

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Keyword (A String Of Length Atleast As Long As Plain Text):");
    if(indent==1)System.out.print("\t\t\t");
    key=bfrd.readLine();
    keyl=key.length();
    if(keyl<ctl)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Key Length");
    }
    else
    {
        temp=0;
    }
} while(temp==1);

for(i=0;i<ctl;i++)
{
    ctc[i]=ct.charAt(i);
    cti[i]=ctc[i];
}

for(i=0;i<keyl;i++)
{
    keyc[i]=key.charAt(i);
```

```
        keyi[i]=keyc[i];
        keyi[i]=keyi[i]-32;
    }

    for(i=0;i<ctl;i++)
    {
        pti[i]=cti[i]-32;
        pti[i]=pti[i]-keyi[i]%95;
        if(pti[i]<0)
        {
            pti[i]=pti[i]+95;
        }
        pti[i]=pti[i]%95;
        pti[i]=pti[i]+32;
    }

    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("The Plain Text Is:");
    if(indent==1)System.out.print("\t\t\t");
    System.out.print("");
    for(i=0;i<ctl;i++)
    {
        ptc[i]=(char)pti[i];
        System.out.print(ptc[i]);
    }
    System.out.println("");
}

public static void MonoalphabeticSubstitution_enc()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int ptl;
    int keyl1;
    int keyl2;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int keyi1[];
    keyi1 = new int[10000];

    int keyi2[];
    keyi2 = new int[10000];

    char ptc[];
    ptc = new char[10000];
```

```
char ctc[];
ctc = new char[10000];

char keyc[];
keyc = new char[10000];

String pt;
String key;

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Plain Text (A String Of Length Greater Than 2):");
    if(indent==1)System.out.print("\t\t\t");
    pt=bfrd.readLine();
    ptl=pt.length();
    if(ptl<3)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Plain Text Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Keyword (A String Of Length Greater Than 1):");
    if(indent==1)System.out.print("\t\t\t");
    key=bfrd.readLine();
    keyl1=key.length();
    if(keyl1<2)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Key Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<ptl;i++)
```



```
{
    ptc[i]=pt.charAt(i);
    pti[i]=ptc[i];
    pti[i]=pti[i]-32;
}

for(i=0;i<keyl1;i++)
{
    keyc[i]=key.charAt(i);
    keyi1[i]=keyc[i];
}

keyl2=0;
for(i=0;i<keyl1;i++)
{
    temp=0;
    for(j=0;j<keyl2;j++)
    {
        if(keyi1[i]==keyi2[j])
        {
            temp=1;
        }
    }
    if(temp==0)
    {
        keyi2[keyl2]=keyi1[i];
        keyl2=keyl2+1;
    }
}

for(i=32;i<=126;i++)
{
    temp=0;
    for(j=0;j<keyl2;j++)
    {
        if(keyi2[j]==i)
        {
            temp=1;
        }
    }
    if(temp==0)
    {
        keyi2[keyl2]=i;
        keyl2=keyl2+1;
    }
}

for(i=0;i<ptl;i++)
{
    cti[i]=keyi2[pti[i]];
}

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
```

```
System.out.println("The Cipher Text Is:");
if(indent==1)System.out.print("\t\t");
System.out.print("");
for(i=0;i<ptl;i++)
{
    ctc[i]=(char)cti[i];
    System.out.print(ctc[i]);
}
System.out.println("");
}

public static void MonoalphabeticSubstitution_dec()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int ctl;
    int key11;
    int key12;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int keyi1[];
    keyi1 = new int[10000];

    int keyi2[];
    keyi2 = new int[10000];

    char ptc[];
    ptc = new char[10000];

    char ctc[];
    ctc = new char[10000];

    char keyc[];
    keyc = new char[10000];

    String ct;
    String key;

    temp=0;
    do
    {
        if(indent==1)System.out.print("\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t");
        System.out.println("Enter The Cipher Text (A String Of Length Greater Than 2):");
        if(indent==1)System.out.print("\t\t");
        ct=bfrd.readLine();
```

```
        ctl=ct.length();
        if(ctl<3)
        {
            temp=1;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Invalid Cipher Text Length");
        }
        else
        {
            temp=0;
        }
    }while(temp==1);

    temp=0;
    do
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter The Keyword (A String Of Length Greater Than 1):");
        if(indent==1)System.out.print("\t\t\t");
        key=bfrd.readLine();
        keyl1=key.length();
        if(keyl1<2)
        {
            temp=1;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Invalid Key Length");
        }
        else
        {
            temp=0;
        }
    }while(temp==1);

    for(i=0;i<ctl;i++)
    {
        ctc[i]=ct.charAt(i);
        cti[i]=ctc[i];
    }

    for(i=0;i<keyl1;i++)
    {
        keyc[i]=key.charAt(i);
        keyi1[i]=keyc[i];
    }

    keyl2=0;
    for(i=0;i<keyl1;i++)
    {
        temp=0;
        for(j=0;j<keyl2;j++)
        {
            if(keyi1[i]==keyi2[j])
            {
```

```
        temp=1;
    }
}
if(temp==0)
{
    keyi2[keyl2]=keyi1[i];
    keyl2=keyl2+1;
}
}

for(i=32;i<=126;i++)
{
    temp=0;
    for(j=0;j<keyl2;j++)
    {
        if(keyi2[j]==i)
        {
            temp=1;
        }
    }
    if(temp==0)
    {
        keyi2[keyl2]=i;
        keyl2=keyl2+1;
    }
}

for(i=0;i<ctl;i++)
{
    for(j=0;j<keyl2;j++)
    {
        if(keyi2[j]==cti[i])
        {
            pti[i]=j;
            pti[i]=pti[i]+32;
        }
    }
}

if(indent==1)System.out.print("\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t");
System.out.println("The Plain Text Is:");
if(indent==1)System.out.print("\t\t");
System.out.print("");
for(i=0;i<ctl;i++)
{
    ptc[i]=(char)pti[i];
    System.out.print(ptc[i]);
}
System.out.println("");
}

public static void PolyalphabeticSubstitution_enc()throws IOException
{
```

```
BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

int i;
int ptl;
int keyl;
int temp;

int pti[];
pti = new int[10000];

int cti[];
cti = new int[10000];

int keyi[];
keyi = new int[10000];

char ptc[];
ptc = new char[10000];

char ctc[];
ctc = new char[10000];

char keyc[];
keyc = new char[10000];

String pt;
String key;

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Plain Text (A String Of Length Greater Than 2:");
    if(indent==1)System.out.print("\t\t\t");
    pt=bfrd.readLine();
    ptl=pt.length();
    if(ptl<3)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Plain Text Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
```

```
if(indent==1)System.out.print("\t\t\t");
System.out.println("Enter The Keyword (A String Of Length Greater Than 1):");
if(indent==1)System.out.print("\t\t\t");
key=bfrd.readLine();
keyl=key.length();
if(keyl<2)
{
    temp=1;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Invalid Key Length");
}
else
{
    temp=0;
}
}while(temp==1);

for(i=0;i<ptl;i++)
{
    ptc[i]=pt.charAt(i);
    pti[i]=ptc[i];
}

for(i=0;i<keyl;i++)
{
    keyc[i]=key.charAt(i);
    keyi[i]=keyc[i];
    keyi[i]=keyi[i]-32;
}

for(i=0;i<ptl;i++)
{
    keyi[i]=keyi[i%keyl];
}

for(i=0;i<ptl;i++)
{
    cti[i]=pti[i]-32;
    cti[i]=cti[i]+keyi[i];
    cti[i]=cti[i]%95;
    cti[i]=cti[i]+32;
}

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Cipher Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<ptl;i++)
{
    ctc[i]=(char)cti[i];
    System.out.print(ctc[i]);
}
System.out.println("");
```

```
}

public static void PolyalphabeticSubstitution_dec()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int ctl;
    int keyl;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int keyi[];
    keyi = new int[10000];

    char ptc[];
    ptc = new char[10000];

    char ctc[];
    ctc = new char[10000];

    char keyc[];
    keyc = new char[10000];

    String ct;
    String key;

    temp=0;
    do
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter The Cipher Text (A String Of Length Greater Than 2):");
        if(indent==1)System.out.print("\t\t\t");
        ct=bfrd.readLine();
        ctl=ct.length();
        if(ctl<3)
        {
            temp=1;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Invalid Cipher Text Length");
        }
        else
        {
            temp=0;
        }
    }while(temp==1);

    temp=0;
```

```
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Keyword (A String Of Length Greater Than 1):");
    if(indent==1)System.out.print("\t\t\t");
    key=bfrd.readLine();
    keyl=key.length();
    if(keyl<2)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Key Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<ctl;i++)
{
    ctc[i]=ct.charAt(i);
    cti[i]=ctc[i];
}

for(i=0;i<keyl;i++)
{
    keyc[i]=key.charAt(i);
    keyi[i]=keyc[i];
    keyi[i]=keyi[i]-32;
}

for(i=0;i<ctl;i++)
{
    keyi[i]=keyi[i%keyl];
}

for(i=0;i<ctl;i++)
{
    pti[i]=cti[i]-32;
    pti[i]=pti[i]-keyi[i]%95;
    if(pti[i]<0)
    {
        pti[i]=pti[i]+95;
    }
    pti[i]=pti[i]%95;
    pti[i]=pti[i]+32;
}

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Plain Text Is:");
```



```
if(indent==1)System.out.print("\t\t");
System.out.print("");
for(i=0;i<ctl;i++)
{
    ptc[i]=(char)pti[i];
    System.out.print(ptc[i]);
}
System.out.println("");
}

public static void Autokey_enc()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int ptl;
    int keyl;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int keyi[];
    keyi = new int[10000];

    char ptc[];
    ptc = new char[10000];

    char ctc[];
    ctc = new char[10000];

    char keyc[];
    keyc = new char[10000];

    String pt;
    String key;

    temp=0;
    do
    {
        if(indent==1)System.out.print("\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t");
        System.out.println("Enter The Plain Text (A String Of Length Greater Than 2):");
        if(indent==1)System.out.print("\t\t");
        pt=bfrd.readLine();
        ptl=pt.length();
        if(ptl<3)
        {
            temp=1;
            if(indent==1)System.out.print("\t\t");
            System.out.println("Invalid Plain Text Length");
        }
    }
```

```
    }
    else
    {
        temp=0;
    }
}while(temp==1);

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Keyword (A String Of Length Greater Than 1):");
    if(indent==1)System.out.print("\t\t\t");
    key=bfrd.readLine();
    keyl=key.length();
    if(keyl<2)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Key Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<ptl;i++)
{
    ptc[i]=pt.charAt(i);
    pti[i]=ptc[i];
}

for(i=0;i<keyl;i++)
{
    keyc[i]=key.charAt(i);
    keyi[i]=keyc[i];
    keyi[i]=keyi[i]-32;
}

for(i=0;i<ptl;i++)
{
    keyi[keyl+i]=pti[i]-32;
}

for(i=0;i<ptl;i++)
{
    cti[i]=pti[i]-32;
    cti[i]=cti[i]+keyi[i];
    cti[i]=cti[i]%95;
    cti[i]=cti[i]+32;
}
```

```
if(indent==1)System.out.print("\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t");
System.out.println("The Cipher Text Is:");
if(indent==1)System.out.print("\t\t");
System.out.print("");
for(i=0;i<ptl;i++)
{
    ctc[i]=(char)cti[i];
    System.out.print(ctc[i]);
}
System.out.println("");
}

public static void Autokey_dec()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int ctl;
    int keyl;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int keyi[];
    keyi = new int[10000];

    char ptc[];
    ptc = new char[10000];

    char ctc[];
    ctc = new char[10000];

    char keyc[];
    keyc = new char[10000];

    String ct;
    String key;

    temp=0;
    do
    {
        if(indent==1)System.out.print("\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t");
        System.out.println("Enter The Cipher Text (A String Of Length Greater Than 2):");
        if(indent==1)System.out.print("\t\t");
        ct=bfrd.readLine();
        ctl=ct.length();
        if(ctl<3)
```

```
{
    temp=1;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Invalid Cipher Text Length");
}
else
{
    temp=0;
}
}while(temp==1);

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Keyword (A String Of Length Greater Than 1):");
    if(indent==1)System.out.print("\t\t\t");
    key=bfrd.readLine();
    keyl=key.length();
    if(keyl<2)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Key Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<ctl;i++)
{
    ctc[i]=ct.charAt(i);
    cti[i]=ctc[i];
}

for(i=0;i<keyl;i++)
{
    keyc[i]=key.charAt(i);
    keyi[i]=keyc[i];
    keyi[i]=keyi[i]-32;
}

for(i=0;i<ctl;i++)
{
    pti[i]=cti[i]-32;
    pti[i]=pti[i]-keyi[i]%95;
    if(pti[i]<0)
    {
        pti[i]=pti[i]+95;
    }
    pti[i]=pti[i]%95;
```

```
        keyi[i+keyl]=pti[i];
        pti[i]=pti[i]+32;
    }

    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("The Plain Text Is:");
    if(indent==1)System.out.print("\t\t\t");
    System.out.print("");
    for(i=0;i<ctl;i++)
    {
        ptc[i]=(char)pti[i];
        System.out.print(ptc[i]);
    }
    System.out.println("");
}

public static void RunningKey_enc()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int ptl;
    int keyl;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int keyi[];
    keyi = new int[10000];

    char ptc[];
    ptc = new char[10000];

    char ctc[];
    ctc = new char[10000];

    char keyc[];
    keyc = new char[10000];

    String pt;
    String key;

    temp=0;
    do
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter The Plain Text (A String Of Length Greater Than 2):");
```

```
if(indent==1)System.out.print("\t\t\t");
pt=bfrd.readLine();
ptl=pt.length();
if(ptl<3)
{
    temp=1;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Invalid Plain Text Length");
}
else
{
    temp=0;
}
}while(temp==1);

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Keyword (A String Of Length Atleast As Long As Plain Text):");
    if(indent==1)System.out.print("\t\t\t");
    key=bfrd.readLine();
    keyl=key.length();
    if(keyl<ptl)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Key Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<ptl;i++)
{
    ptc[i]=pt.charAt(i);
    pti[i]=ptc[i];
}

for(i=0;i<keyl;i++)
{
    keyc[i]=key.charAt(i);
    keyi[i]=keyc[i];
    keyi[i]=keyi[i]-32;
}

for(i=0;i<ptl;i++)
{
    cti[i]=pti[i]-32;
    cti[i]=cti[i]+keyi[i];
    cti[i]=cti[i]%95;
```

```
        cti[i]=cti[i]+32;
    }

    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("The Cipher Text Is:");
    if(indent==1)System.out.print("\t\t\t");
    System.out.print("");
    for(i=0;i<ptl;i++)
    {
        ctc[i]=(char)cti[i];
        System.out.print(ctc[i]);
    }
    System.out.println("");
}

public static void RunningKey_dec()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int ctl;
    int keyl;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int keyi[];
    keyi = new int[10000];

    char ptc[];
    ptc = new char[10000];

    char ctc[];
    ctc = new char[10000];

    char keyc[];
    keyc = new char[10000];

    String ct;
    String key;

    temp=0;
    do
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter The Cipher Text (A String Of Length Greater Than 2):");
        if(indent==1)System.out.print("\t\t\t");
```

```
ct=bfrd.readLine();
ctl=ct.length();
if(ctl<3)
{
    temp=1;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Invalid Cipher Text Length");
}
else
{
    temp=0;
}
}while(temp==1);

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Keyword (A String Of Length Atleast As Long As Plain Text):");
    if(indent==1)System.out.print("\t\t\t");
    key=bfrd.readLine();
    keyl=key.length();
    if(keyl<ctl)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Key Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<ctl;i++)
{
    ctc[i]=ct.charAt(i);
    cti[i]=ctc[i];
}

for(i=0;i<keyl;i++)
{
    keyc[i]=key.charAt(i);
    keyi[i]=keyc[i];
    keyi[i]=keyi[i]-32;
}

for(i=0;i<ctl;i++)
{
    pti[i]=cti[i]-32;
    pti[i]=pti[i]-keyi[i]%95;
    if(pti[i]<0)
    {
```



```
        pti[i]=pti[i]+95;
    }
    pti[i]=pti[i]%95;
    pti[i]=pti[i]+32;
}

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Plain Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<ctl;i++)
{
    ptc[i]=(char)pti[i];
    System.out.print(ptc[i]);
}
System.out.println("");
}

public static void PolybiusSquare_enc()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int ptl;
    int ctl;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int polybiussquare[][];
    polybiussquare = new int[10][10];

    char ptc[];
    ptc = new char[10000];

    String pt;

    temp=0;
    do
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter The Plain Text (A String Of Length Greater Than 2):");
        if(indent==1)System.out.print("\t\t\t");
        pt=bfrd.readLine();
        ptl=pt.length();
```

```
    if(ptl<3)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Plain Text Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<ptl;i++)
{
    ptc[i]=pt.charAt(i);
    pti[i]=ptc[i];
}

k=0;
for(i=0;i<10;i++)
{
    for(j=0;j<10;j++)
    {
        if(k<95)
        {
            polybiussquare[i][j]=k+32;
            k=k+1;
        }
        else
        {
            polybiussquare[i][j]=-1;
        }
    }
}

ctl=0;
for(k=0;k<ptl;k++)
{
    temp=0;
    for(i=0;i<10;i++)
    {
        for(j=0;j<10;j++)
        {
            if(temp==0)
            {
                if(pti[k]==polybiussquare[i][j])
                {
                    temp=1;
                    cti[ctl]=i;
                    ctl=ctl+1;
                    cti[ctl]=j;
                    ctl=ctl+1;;
                }
            }
        }
    }
}
```

```

    }
}

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Cipher Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<ctl;i++)
{
    System.out.print(cti[i]);
}
System.out.println("");
}

public static void PolybiusSquare_dec()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int pti;
    int ctl;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int polybiussquare[][];
    polybiussquare = new int[10][10];

    char ptc[];
    ptc = new char[10000];

    char ctc[];
    ctc = new char[10000];

    String ct;

    temp=0;
    do
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter The Cipher Text (A Sequence Of Numbers Of Length Greater Than 4 And
Of Length A Multiple Of 2):");
        if(indent==1)System.out.print("\t\t\t");
        ct=bfrd.readLine();
        ctl=ct.length();

```

```
if(ctl<5||(ctl%2)!=0)
{
    temp=1;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Invalid Cipher Text Length");
}

else
{
    temp=0;
    for(i=0;i<ctl;i++)
    {
        ctc[i]=ct.charAt(i);
        cti[i]=ctc[i];
    }

    for(i=0;i<ctl;i++)
    {
        if(cti[i]<48||cti[i]>57)
        {
            temp=1;
        }
    }

    if(temp==1)
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Cipher Text");
    }
}
}while(temp==1);

k=0;
for(i=0;i<10;i++)
{
    for(j=0;j<10;j++)
    {
        if(k<95)
        {
            polybiussquare[i][j]=k+32;
            k=k+1;
        }
        else
        {
            polybiussquare[i][j]=-1;
        }
    }
}

ptl=ctl/2;
for(k=0;k<ptl;k++)
{
    temp=k*2;
    i=cti[temp]-48;
```

```
j=cti[temp+1]-48;
pti[k]=polybiussquare[i][j];
}

if(indent==1)System.out.print("\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t");
System.out.println("The Plain Text Is:");
if(indent==1)System.out.print("\t\t");
System.out.print("");
for(i=0;i<ptl;i++)
{
    ptc[i]=(char)pti[i];
    System.out.print(ptc[i]);
}
System.out.println("");
}

public static void Bifid_enc()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int ptl;
    int ctl;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int polybiussquare[][];
    polybiussquare = new int[10][10];

    char ptc[];
    ptc = new char[10000];

    String pt;

    temp=0;
    do
    {
        if(indent==1)System.out.print("\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t");
        System.out.println("Enter The Plain Text (A String Of Length Greater Than 2):");
        if(indent==1)System.out.print("\t\t");
        pt=bfrd.readLine();
        ptl=pt.length();
        if(ptl<3)
        {
```

```
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Plain Text Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<ptl;i++)
{
    ptc[i]=pt.charAt(i);
    pti[i]=ptc[i];
}

k=0;
for(i=0;i<10;i++)
{
    for(j=0;j<10;j++)
    {
        if(k<95)
        {
            polybiussquare[i][j]=k+32;
            k=k+1;
        }
        else
        {
            polybiussquare[i][j]=-1;
        }
    }
}

ctl=0;
for(k=0;k<ptl;k++)
{
    temp=0;
    for(i=0;i<10;i++)
    {
        for(j=0;j<10;j++)
        {
            if(temp==0)
            {
                if(pti[k]==polybiussquare[i][j])
                {
                    temp=1;
                    cti[ctl]=i;
                    cti[ctl+ptl]=j;
                    ctl=ctl+1;
                }
            }
        }
    }
}

ctl=ptl*2;
```

```

        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("The Cipher Text Is:");
        if(indent==1)System.out.print("\t\t\t");
        System.out.print("");
        for(i=0;i<ctl;i++)
        {
            System.out.print(cti[i]);
        }
        System.out.println("");
    }

    public static void Bifid_dec()throws IOException
    {
        BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

        int i;
        int j;
        int k;
        int ptl;
        int ctl;
        int temp;

        int pti[];
        pti = new int[10000];

        int cti[];
        cti = new int[10000];

        int polybiussquare[][];
        polybiussquare = new int[10][10];

        char ptc[];
        ptc = new char[10000];

        char ctc[];
        ctc = new char[10000];

        String ct;

        temp=0;
        do
        {
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Enter The Cipher Text (A Sequence Of Numbers Of Length Greater Than 4 And
Of Length A Multiple Of 2):");
            if(indent==1)System.out.print("\t\t\t");
            ct=bfrd.readLine();
            ctl=ct.length();

            if(ctl<5||(ctl%2)!=0)

```

```
{
    temp=1;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Invalid Cipher Text Length");
}

else
{
    temp=0;
    for(i=0;i<ctl;i++)
    {
        ctc[i]=ct.charAt(i);
        cti[i]=ctc[i];
    }

    for(i=0;i<ctl;i++)
    {
        if(cti[i]<48||cti[i]>57)
        {
            temp=1;
        }
    }

    if(temp==1)
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Cipher Text");
    }
}
}while(temp==1);

k=0;
for(i=0;i<10;i++)
{
    for(j=0;j<10;j++)
    {
        if(k<95)
        {
            polybiussquare[i][j]=k+32;
            k=k+1;
        }
        else
        {
            polybiussquare[i][j]=-1;
        }
    }
}

ptl=ctl/2;
for(k=0;k<ptl;k++)
{
    i=cti[k]-48;
    j=cti[k+ptl]-48;
    pti[k]=polybiussquare[i][j];
}
```



```
if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Plain Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<ptl;i++)
{
    ptc[i]=(char)pti[i];
    System.out.print(ptc[i]);
}
System.out.println("");
}

public static void Trifid_enc()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int l;
    int ptl;
    int ctl;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int trifidcube[][][];
    trifidcube = new int[3][6][6];

    char ptc[];
    ptc = new char[10000];

    String pt;

    temp=0;
    do
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter The Plain Text (A String Of Length Greater Than 2):");
        if(indent==1)System.out.print("\t\t\t");
        pt=bfrd.readLine();
        ptl=pt.length();
        if(ptl<3)
        {
            temp=1;
            if(indent==1)System.out.print("\t\t\t");
```

```
        System.out.println("Invalid Plain Text Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<ptl;i++)
{
    ptc[i]=pt.charAt(i);
    pti[i]=ptc[i];
}

l=0;
for(i=0;i<3;i++)
{
    for(j=0;j<6;j++)
    {
        for(k=0;k<6;k++)
        {
            if(l<95)
            {
                trificube[i][j][k]=l+32;
                l=l+1;
            }
            else
            {
                trificube[i][j][k]=-1;
            }
        }
    }
}

for(l=0;l<ptl;l++)
{
    temp=0;
    for(i=0;i<3;i++)
    {
        for(j=0;j<6;j++)
        {
            for(k=0;k<6;k++)
            {
                if(temp==0);
                {
                    if(pti[l]==trificube[i][j][k])
                    {
                        temp=1;
                        cti[l]=i;
                        cti[l+ptl]=j;
                        cti[l+ptl+ptl]=k;
                    }
                }
            }
        }
    }
}
```

```
    }
}

ctl=ptl*3;

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Cipher Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<ctl;i++)
{
    System.out.print(cti[i]);
}
System.out.println("");
}

public static void Trifid_dec()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int l;
    int ptl;
    int ctl;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int trifidcube[][][];
    trifidcube = new int[3][6][6];

    char ptc[];
    ptc = new char[10000];

    char ctc[];
    ctc = new char[10000];

    String ct;

    temp=0;
    do
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter The Cipher Text (A Sequence Of Numbers Of Length Greater Than 6 And
Of Length A Multiple Of 3):");
```

```
if(indent==1)System.out.print("\t\t\t");
ct=bfrd.readLine();
ctl=ct.length();

if(ctl<7||(ctl%3)!=0)
{
    temp=1;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Invalid Cipher Text Length");
}

else
{
    temp=0;
    for(i=0;i<ctl;i++)
    {
        ctc[i]=ct.charAt(i);
        cti[i]=ctc[i];
    }

    for(i=0;i<ctl;i++)
    {
        if(cti[i]<48||cti[i]>57)
        {
            temp=1;
        }
    }

    if(temp==1)
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Cipher Text");
    }
}
}while(temp==1);

l=0;
for(i=0;i<3;i++)
{
    for(j=0;j<6;j++)
    {
        for(k=0;k<6;k++)
        {
            if(l<95)
            {
                trificube[i][j][k]=l+32;
                l=l+1;
            }
            else
            {
                trificube[i][j][k]=-1;
            }
        }
    }
}
```

```
    ptl=ctl/3;
    for(l=0;l<ptl;l++)
    {
        i=cti[l]-48;
        j=cti[l+ptl]-48;
        k=cti[l+ptl+ptl]-48;
        pti[l]=trifidcube[i][j][k];
    }

    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("The Plain Text Is:");
    if(indent==1)System.out.print("\t\t\t");
    System.out.print("");
    for(i=0;i<ptl;i++)
    {
        ptc[i]=(char)pti[i];
        System.out.print(ptc[i]);
    }
    System.out.println("");
}

public static void BitwiseRotation_enc()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int pow;
    int ptl;
    int digits;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int ptbinary[][];
    ptbinary = new int[10000][8];

    int tempbinary[];
    tempbinary = new int[8];

    char ptc[];
    ptc = new char[10000];

    String pt;

    temp=0;
    do
```

```
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Plain Text (A String Of Length Greater Than 2):");
    if(indent==1)System.out.print("\t\t\t");
    pt=bfrd.readLine();
    ptl=pt.length();
    if(ptl<3)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Plain Text Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<ptl;i++)
{
    ptc[i]=pt.charAt(i);
    pti[i]=ptc[i];
}

for(i=0;i<ptl;i++)
{
    temp=pti[i];
    for(j=7;j>=0;j--)
    {
        pow=1;
        for(k=1;k<=j;k++)
        {
            pow=pow*2;
        }

        if(pow<=temp)
        {
            ptbinary[i][7-j]=1;
            temp=temp-pow;
        }
        else
        {
            ptbinary[i][7-j]=0;
        }
    }
}

for(i=0;i<ptl;i++)
{
    for(j=0;j<=7;j++)
    {
        tempbinary[j]=ptbinary[i][j];
    }
}
```

```
    ptbinary[i][0]=tempbinary[4];
    ptbinary[i][1]=tempbinary[5];
    ptbinary[i][2]=tempbinary[6];
    ptbinary[i][3]=tempbinary[7];
    ptbinary[i][4]=tempbinary[0];
    ptbinary[i][5]=tempbinary[1];
    ptbinary[i][6]=tempbinary[2];
    ptbinary[i][7]=tempbinary[3];
}

for(i=0;i<ptl;i++)
{
    temp=0;
    pow=1;
    for(j=7;j>=0;j--)
    {
        temp=temp+pow*ptbinary[i][j];
        pow=pow*2;
    }
    cti[i]=temp;
}

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Cipher Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<ptl;i++)
{
    temp=cti[i];
    digits=0;
    do
    {
        digits=digits+1;
        temp=temp/10;
    }while(temp!=0);
    if(digits==0)
    {
        System.out.print("000");
    }
    else if(digits==1)
    {
        System.out.print("00");
    }
    else if(digits==2)
    {
        System.out.print("0");
    }
    else if(digits==3)
    {
        System.out.print("");
    }
    System.out.print(cti[i]);
}
```

```
        System.out.println("");
    }

    public static void BitwiseRotation_dec()throws IOException
    {
        BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

        int i;
        int j;
        int k;
        int l;
        int pow;
        int ctl;
        int ctipl;
        int temp;

        int pti[];
        pti = new int[10000];

        int cti[];
        cti = new int[10000];

        int ctipi[];
        ctipi = new int[30000];

        int ctbinary[][];
        ctbinary = new int[10000][8];

        int tempbinary[];
        tempbinary = new int[8];

        char ptc[];
        ptc = new char[10000];

        char ctc[];
        ctc = new char[10000];

        char ctipc[];
        ctipc = new char[30000];

        String ct;

        temp=0;
        do
        {
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Enter The Cipher Text (A Sequence Of Numbers Of Length Greater Than 6 And
Of Length A Multiple Of 3):");
            if(indent==1)System.out.print("\t\t\t");
            ct=bfrd.readLine();
            ctipl=ct.length();

            if(ctipl<7||(ctipl%3)!=0)
```



```
{
    temp=1;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Invalid Cipher Text Length");
}

else
{
    temp=0;
    for(i=0;i<ctipl;i++)
    {
        ctipc[i]=ct.charAt(i);
        ctipi[i]=ctipc[i];
    }

    for(i=0;i<ctipl;i++)
    {
        if(ctipi[i]<48||ctipi[i]>57)
        {
            temp=1;
        }
    }

    if(temp==1)
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Cipher Text");
    }
}
}while(temp==1);

ctl=ctipl/3;

for(i=0;i<ctl;i++)
{
    j=i*3;
    cti[i]=0;
    for(k=0;k<3;k++)
    {
        if(k==0)
        {
            l=100;
        }
        else if(k==1)
        {
            l=10;
        }
        else
        {
            l=1;
        }
        cti[i]=cti[i]+(ctipi[j+k]-48)*l;
    }
}
```

```
for(i=0;i<ctl;i++)
{
    temp=cti[i];
    for(j=7;j>=0;j--)
    {
        pow=1;
        for(k=1;k<=j;k++)
        {
            pow=pow*2;
        }

        if(pow<=temp)
        {
            ctbinary[i][7-j]=1;
            temp=temp-pow;
        }
        else
        {
            ctbinary[i][7-j]=0;
        }
    }
}

for(i=0;i<ctl;i++)
{
    for(j=0;j<=7;j++)
    {
        tempbinary[j]=ctbinary[i][j];
    }
    ctbinary[i][0]=tempbinary[4];
    ctbinary[i][1]=tempbinary[5];
    ctbinary[i][2]=tempbinary[6];
    ctbinary[i][3]=tempbinary[7];
    ctbinary[i][4]=tempbinary[0];
    ctbinary[i][5]=tempbinary[1];
    ctbinary[i][6]=tempbinary[2];
    ctbinary[i][7]=tempbinary[3];
}

for(i=0;i<ctl;i++)
{
    temp=0;
    pow=1;
    for(j=7;j>=0;j--)
    {
        temp=temp+pow*ctbinary[i][j];
        pow=pow*2;
    }
    pti[i]=temp;
}

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Plain Text Is:");
```

```
if(indent==1)System.out.print("\t\t");
System.out.print("");
for(i=0;i<cti;i++)
{
    ptc[i]=(char)pti[i];
    System.out.print(ptc[i]);
}
System.out.println("");
}

public static void BitwiseAtbash_enc()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int pow;
    int pti;
    int digits;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int ptbinary[][];
    ptbinary = new int[10000][8];

    int tempbinary[];
    tempbinary = new int[8];

    char ptc[];
    ptc = new char[10000];

    String pt;

    temp=0;
    do
    {
        if(indent==1)System.out.print("\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t");
        System.out.println("Enter The Plain Text (A String Of Length Greater Than 2):");
        if(indent==1)System.out.print("\t\t");
        pt=bfrd.readLine();
        pti=pt.length();
        if(pti<3)
        {
            temp=1;
            if(indent==1)System.out.print("\t\t");
            System.out.println("Invalid Plain Text Length");
        }
    }
```

```
        else
        {
            temp=0;
        }
    }while(temp==1);

    for(i=0;i<ptl;i++)
    {
        ptc[i]=pt.charAt(i);
        pti[i]=ptc[i];
    }

    for(i=0;i<ptl;i++)
    {
        temp=pti[i];
        for(j=7;j>=0;j--)
        {
            pow=1;
            for(k=1;k<=j;k++)
            {
                pow=pow*2;
            }

            if(pow<=temp)
            {
                ptbinary[i][7-j]=1;
                temp=temp-pow;
            }
            else
            {
                ptbinary[i][7-j]=0;
            }
        }
    }

    for(i=0;i<ptl;i++)
    {
        for(j=0;j<=7;j++)
        {
            tempbinary[j]=ptbinary[i][j];
        }
        ptbinary[i][0]=tempbinary[7];
        ptbinary[i][1]=tempbinary[6];
        ptbinary[i][2]=tempbinary[5];
        ptbinary[i][3]=tempbinary[4];
        ptbinary[i][4]=tempbinary[3];
        ptbinary[i][5]=tempbinary[2];
        ptbinary[i][6]=tempbinary[1];
        ptbinary[i][7]=tempbinary[0];
    }

    for(i=0;i<ptl;i++)
    {
        temp=0;
        pow=1;
```

```
        for(j=7;j>=0;j--)
        {
            temp=temp+pow*ptbinary[i][j];
            pow=pow*2;
        }
        cti[i]=temp;
    }

    if(indent==1)System.out.print("\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t");
    System.out.println("The Cipher Text Is:");
    if(indent==1)System.out.print("\t\t");
    System.out.print("");
    for(i=0;i<ptl;i++)
    {
        temp=cti[i];
        digits=0;
        do
        {
            digits=digits+1;
            temp=temp/10;
        }while(temp!=0);
        if(digits==0)
        {
            System.out.print("000");
        }
        else if(digits==1)
        {
            System.out.print("00");
        }
        else if(digits==2)
        {
            System.out.print("0");
        }
        else if(digits==3)
        {
            System.out.print("");
        }
        System.out.print(cti[i]);
    }
    System.out.println("");
}

public static void BitwiseAtbash_dec()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int l;
    int pow;
    int cti;
    int ctipl;
```

```
int temp;

int pti[];
pti = new int[10000];

int cti[];
cti = new int[10000];

int ctipi[];
ctipi = new int[30000];

int ctbinary[][];
ctbinary = new int[10000][8];

int tempbinary[];
tempbinary = new int[8];

char ptc[];
ptc = new char[10000];

char ctc[];
ctc = new char[10000];

char ctipc[];
ctipc = new char[30000];

String ct;

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Cipher Text (A Sequence Of Numbers Of Length Greater Than 6 And
Of Length A Multiple Of 3):");
    if(indent==1)System.out.print("\t\t\t");
    ct=bfrd.readLine();
    ctipl=ct.length();

    if(ctipl<7||(ctipl%3)!=0)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Cipher Text Length");
    }

    else
    {
        temp=0;
        for(i=0;i<ctipl;i++)
        {
            ctipc[i]=ct.charAt(i);
            ctipi[i]=ctipc[i];
        }
    }
}
```

```
        for(i=0;i<ctipl;i++)
        {
            if(ctipi[i]<48||ctipi[i]>57)
            {
                temp=1;
            }
        }

        if(temp==1)
        {
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Invalid Cipher Text");
        }
    }
}while(temp==1);

ctl=ctipl/3;

for(i=0;i<ctl;i++)
{
    j=i*3;
    cti[i]=0;
    for(k=0;k<3;k++)
    {
        if(k==0)
        {
            l=100;
        }
        else if(k==1)
        {
            l=10;
        }
        else
        {
            l=1;
        }
        cti[i]=cti[i]+(ctipi[j+k]-48)*l;
    }
}

for(i=0;i<ctl;i++)
{
    temp=cti[i];
    for(j=7;j>=0;j--)
    {
        pow=1;
        for(k=1;k<=j;k++)
        {
            pow=pow*2;
        }

        if(pow<=temp)
        {
            ctbinary[i][7-j]=1;
        }
    }
}
```

```
        temp=temp-pow;
    }
    else
    {
        ctbinary[i][7-j]=0;
    }
}
}

for(i=0;i<ctl;i++)
{
    for(j=0;j<=7;j++)
    {
        tempbinary[j]=ctbinary[i][j];
    }
    ctbinary[i][0]=tempbinary[7];
    ctbinary[i][1]=tempbinary[6];
    ctbinary[i][2]=tempbinary[5];
    ctbinary[i][3]=tempbinary[4];
    ctbinary[i][4]=tempbinary[3];
    ctbinary[i][5]=tempbinary[2];
    ctbinary[i][6]=tempbinary[1];
    ctbinary[i][7]=tempbinary[0];
}

for(i=0;i<ctl;i++)
{
    temp=0;
    pow=1;
    for(j=7;j>=0;j--)
    {
        temp=temp+pow*ctbinary[i][j];
        pow=pow*2;
    }
    pti[i]=temp;
}

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Plain Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<ctl;i++)
{
    ptc[i]=(char)pti[i];
    System.out.print(ptc[i]);
}
System.out.println("");
}

public static void BitwiseBinaryCompliment_enc()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));
```



```
int i;
int j;
int k;
int pow;
int ptl;
int digits;
int temp;

int pti[];
pti = new int[10000];

int cti[];
cti = new int[10000];

int ptbinary[][];
ptbinary = new int[10000][8];

char ptc[];
ptc = new char[10000];

String pt;

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Plain Text (A String Of Length Greater Than 2):");
    if(indent==1)System.out.print("\t\t\t");
    pt=bfrd.readLine();
    ptl=pt.length();
    if(ptl<3)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Plain Text Length");
    }
    else
    {
        temp=0;
    }
} while(temp==1);

for(i=0;i<ptl;i++)
{
    ptc[i]=pt.charAt(i);
    pti[i]=ptc[i];
}

for(i=0;i<ptl;i++)
{
    temp=pti[i];
    for(j=7;j>=0;j--)
    {
```

```
        pow=1;
        for(k=1;k<=j;k++)
        {
            pow=pow*2;
        }

        if(pow<=temp)
        {
            ptbinary[i][7-j]=1;
            temp=temp-pow;
        }
        else
        {
            ptbinary[i][7-j]=0;
        }
    }
}

for(i=0;i<ptl;i++)
{
    for(j=0;j<=7;j++)
    {
        if(ptbinary[i][j]==1)
        {
            ptbinary[i][j]=0;
        }
        else if(ptbinary[i][j]==0)
        {
            ptbinary[i][j]=1;
        }
    }
}

for(i=0;i<ptl;i++)
{
    temp=0;
    pow=1;
    for(j=7;j>=0;j--)
    {
        temp=temp+pow*ptbinary[i][j];
        pow=pow*2;
    }
    cti[i]=temp;
}

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Cipher Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<ptl;i++)
{
    temp=cti[i];
    digits=0;
```

```
do
{
    digits=digits+1;
    temp=temp/10;
}while(temp!=0);
if(digits==0)
{
    System.out.print("000");
}
else if(digits==1)
{
    System.out.print("00");
}
else if(digits==2)
{
    System.out.print("0");
}
else if(digits==3)
{
    System.out.print("");
}
System.out.print(cti[i]);
}
System.out.println("");
}
```

```
public static void BitwiseBinaryCompliment_dec()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int l;
    int pow;
    int ctl;
    int ctipl;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int ctipi[];
    ctipi = new int[30000];

    int ctbinary[][];
    ctbinary = new int[10000][8];

    char ptc[];
    ptc = new char[10000];

    char ctc[];
```

```
ctc = new char[10000];

char ctipc[];
ctipc = new char[30000];

String ct;

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Cipher Text (A Sequence Of Numbers Of Length Greater Than 6 And
Of Length A Multiple Of 3):");
    if(indent==1)System.out.print("\t\t\t");
    ct=bfrd.readLine();
    ctipl=ct.length();

    if(ctipl<7||(ctipl%3)!=0)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Cipher Text Length");
    }

    else
    {
        temp=0;
        for(i=0;i<ctipl;i++)
        {
            ctipc[i]=ct.charAt(i);
            ctipi[i]=ctipc[i];
        }

        for(i=0;i<ctipl;i++)
        {
            if(ctipi[i]<48||ctipi[i]>57)
            {
                temp=1;
            }
        }

        if(temp==1)
        {
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Invalid Cipher Text");
        }
    }
}while(temp==1);

ctl=ctipl/3;

for(i=0;i<ctl;i++)
{
```

```
j=i*3;
cti[i]=0;
for(k=0;k<3;k++)
{
    if(k==0)
    {
        l=100;
    }
    else if(k==1)
    {
        l=10;
    }
    else
    {
        l=1;
    }
    cti[i]=cti[i]+(ctipi[j+k]-48)*l;
}
}

for(i=0;i<ctl;i++)
{
    temp=cti[i];
    for(j=7;j>=0;j--)
    {
        pow=1;
        for(k=1;k<=j;k++)
        {
            pow=pow*2;
        }

        if(pow<=temp)
        {
            ctbinary[i][7-j]=1;
            temp=temp-pow;
        }
        else
        {
            ctbinary[i][7-j]=0;
        }
    }
}

for(i=0;i<ctl;i++)
{
    for(j=0;j<=7;j++)
    {
        if(ctbinary[i][j]==1)
        {
            ctbinary[i][j]=0;
        }
        else if(ctbinary[i][j]==0)
        {
            ctbinary[i][j]=1;
        }
    }
}
```

```
    }
}

for(i=0;i<ctl;i++)
{
    temp=0;
    pow=1;
    for(j=7;j>=0;j--)
    {
        temp=temp+pow*ctbinary[i][j];
        pow=pow*2;
    }
    pti[i]=temp;
}

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Plain Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<ctl;i++)
{
    ptc[i]=(char)pti[i];
    System.out.print(ptc[i]);
}
System.out.println("");
}

public static void BitwiseLeftShift_enc()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int key;
    int pow;
    int pti;
    int digits;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int ptbinary[][];
    ptbinary = new int[10000][8];

    int tempbinary[];
    tempbinary = new int[8];

    char ptc[];
```

```
ptc = new char[10000];

String pt;

key=0;

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Plain Text (A String Of Length Greater Than 2):");
    if(indent==1)System.out.print("\t\t\t");
    pt=bfrd.readLine();
    ptl=pt.length();
    if(ptl<3)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Plain Text Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

do
{
    temp=0;
    while(temp==0)
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter the Key (A Number In The Range Of 0 Through 7):");
        if(indent==1)System.out.print("\t\t\t");
        try
        {
            key=Integer.parseInt(bfrd.readLine());
            temp=1;
        }
        catch(NumberFormatException E1nfe)
        {
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Key Should Be A Number");
            temp=0;
        }
    }

    if(key<0||key>7)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
```

```
        System.out.println("Invalid Key");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<ptl;i++)
{
    ptc[i]=pt.charAt(i);
    pti[i]=ptc[i];
}

for(i=0;i<ptl;i++)
{
    temp=pti[i];
    for(j=7;j>=0;j--)
    {
        pow=1;
        for(k=1;k<=j;k++)
        {
            pow=pow*2;
        }

        if(pow<=temp)
        {
            ptbinary[i][7-j]=1;
            temp=temp-pow;
        }
        else
        {
            ptbinary[i][7-j]=0;
        }
    }
}

for(i=0;i<ptl;i++)
{
    for(j=0;j<=7;j++)
    {
        tempbinary[j]=ptbinary[i][j];
    }
    for(j=0;j<=7;j++)
    {
        ptbinary[i][j]=tempbinary[(j+8+key)%8];
    }
}

for(i=0;i<ptl;i++)
{
    temp=0;
    pow=1;
    for(j=7;j>=0;j--)
    {
```



```
        temp=temp+pow*ptbinary[i][j];
        pow=pow*2;
    }
    cti[i]=temp;
}

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Cipher Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<ptl;i++)
{
    temp=cti[i];
    digits=0;
    do
    {
        digits=digits+1;
        temp=temp/10;
    }while(temp!=0);
    if(digits==0)
    {
        System.out.print("000");
    }
    else if(digits==1)
    {
        System.out.print("00");
    }
    else if(digits==2)
    {
        System.out.print("0");
    }
    else if(digits==3)
    {
        System.out.print("");
    }
    System.out.print(cti[i]);
}
System.out.println("");
}

public static void BitwiseLeftShift_dec()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int l;
    int key;
    int pow;
    int ctl;
    int ctipl;
    int temp;
```

```
int pti[];
pti = new int[10000];

int cti[];
cti = new int[10000];

int ctipi[];
ctipi = new int[30000];

int ctbinary[][];
ctbinary = new int[10000][8];

int tempbinary[];
tempbinary = new int[8];

char ptc[];
ptc = new char[10000];

char ctc[];
ctc = new char[10000];

char ctipc[];
ctipc = new char[30000];

String ct;

key=0;

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Cipher Text (A Sequence Of Numbers Of Length Greater Than 6 And
Of Length A Multiple Of 3):");
    if(indent==1)System.out.print("\t\t\t");
    ct=bfrd.readLine();
    ctipl=ct.length();

    if(ctipl<7||(ctipl%3)!=0)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Cipher Text Length");
    }

    else
    {
        temp=0;
        for(i=0;i<ctipl;i++)
        {
            ctipc[i]=ct.charAt(i);
            ctipi[i]=ctipc[i];
```

```
    }

    for(i=0;i<ctipl;i++)
    {
        if(ctipi[i]<48||ctipi[i]>57)
        {
            temp=1;
        }
    }

    if(temp==1)
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Cipher Text");
    }
}
}while(temp==1);

do
{
    temp=0;
    while(temp==0)
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter the Key (A Number In The Range Of 0 Through 7):");
        if(indent==1)System.out.print("\t\t\t");
        try
        {
            key=Integer.parseInt(bfrd.readLine());
            temp=1;
        }
        catch(NumberFormatException E1nfe)
        {
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Key Should Be A Number");
            temp=0;
        }
    }
}

if(key<0||key>7)
{
    temp=1;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Invalid Key");
}
else
{
    temp=0;
}
}while(temp==1);

ctl=ctipl/3;
```

```
for(i=0;i<ctl;i++)
{
    j=i*3;
    cti[i]=0;
    for(k=0;k<3;k++)
    {
        if(k==0)
        {
            l=100;
        }
        else if(k==1)
        {
            l=10;
        }
        else
        {
            l=1;
        }
        cti[i]=cti[i]+(ctipi[j+k]-48)*l;
    }
}
```

```
for(i=0;i<ctl;i++)
{
    temp=cti[i];
    for(j=7;j>=0;j--)
    {
        pow=1;
        for(k=1;k<=j;k++)
        {
            pow=pow*2;
        }

        if(pow<=temp)
        {
            ctbinary[i][7-j]=1;
            temp=temp-pow;
        }
        else
        {
            ctbinary[i][7-j]=0;
        }
    }
}
```

```
for(i=0;i<ctl;i++)
{
    for(j=0;j<=7;j++)
    {
        tempbinary[j]=ctbinary[i][j];
    }
    for(j=0;j<=7;j++)
    {
        ctbinary[i][j]=tempbinary[(j+8-key)%8];
    }
}
```

```
    }
}

for(i=0;i<ctl;i++)
{
    temp=0;
    pow=1;
    for(j=7;j>=0;j--)
    {
        temp=temp+pow*ctbinary[i][j];
        pow=pow*2;
    }
    pti[i]=temp;
}

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Plain Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<ctl;i++)
{
    ptc[i]=(char)pti[i];
    System.out.print(ptc[i]);
}
System.out.println("");
}

public static void BitwiseRightShift_enc()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int key;
    int pow;
    int ptl;
    int digits;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int ptbinary[][];
    ptbinary = new int[10000][8];

    int tempbinary[];
    tempbinary = new int[8];

    char ptc[];
```

```
ptc = new char[10000];

String pt;

key=0;

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Plain Text (A String Of Length Greater Than 2):");
    if(indent==1)System.out.print("\t\t\t");
    pt=bfrd.readLine();
    ptl=pt.length();
    if(ptl<3)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Plain Text Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

do
{
    temp=0;
    while(temp==0)
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter the Key (A Number In The Range Of 0 Through 7):");
        if(indent==1)System.out.print("\t\t\t");
        try
        {
            key=Integer.parseInt(bfrd.readLine());
            temp=1;
        }
        catch(NumberFormatException E1nfe)
        {
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Key Should Be A Number");
            temp=0;
        }
    }

    if(key<0||key>7)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
```

```
        System.out.println("Invalid Key");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<ptl;i++)
{
    ptc[i]=pt.charAt(i);
    pti[i]=ptc[i];
}

for(i=0;i<ptl;i++)
{
    temp=pti[i];
    for(j=7;j>=0;j--)
    {
        pow=1;
        for(k=1;k<=j;k++)
        {
            pow=pow*2;
        }

        if(pow<=temp)
        {
            ptbinary[i][7-j]=1;
            temp=temp-pow;
        }
        else
        {
            ptbinary[i][7-j]=0;
        }
    }
}

for(i=0;i<ptl;i++)
{
    for(j=0;j<=7;j++)
    {
        tempbinary[j]=ptbinary[i][j];
    }
    for(j=0;j<=7;j++)
    {
        ptbinary[i][j]=tempbinary[(j+8-key)%8];
    }
}

for(i=0;i<ptl;i++)
{
    temp=0;
    pow=1;
    for(j=7;j>=0;j--)
    {
```

```
        temp=temp+pow*ptbinary[i][j];
        pow=pow*2;
    }
    cti[i]=temp;
}

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Cipher Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<ptl;i++)
{
    temp=cti[i];
    digits=0;
    do
    {
        digits=digits+1;
        temp=temp/10;
    }while(temp!=0);
    if(digits==0)
    {
        System.out.print("000");
    }
    else if(digits==1)
    {
        System.out.print("00");
    }
    else if(digits==2)
    {
        System.out.print("0");
    }
    else if(digits==3)
    {
        System.out.print("");
    }
    System.out.print(cti[i]);
}
System.out.println("");
}

public static void BitwiseRightShift_dec()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int l;
    int key;
    int pow;
    int ctl;
    int ctipl;
    int temp;
```



```
int pti[];
pti = new int[10000];

int cti[];
cti = new int[10000];

int ctipi[];
ctipi = new int[30000];

int ctbinary[][];
ctbinary = new int[10000][8];

int tempbinary[];
tempbinary = new int[8];

char ptc[];
ptc = new char[10000];

char ctc[];
ctc = new char[10000];

char ctipc[];
ctipc = new char[30000];

String ct;

key=0;

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Cipher Text (A Sequence Of Numbers Of Length Greater Than 6 And
Of Length A Multiple Of 3):");
    if(indent==1)System.out.print("\t\t\t");
    ct=bfrd.readLine();
    ctipl=ct.length();

    if(ctipl<7||(ctipl%3)!=0)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Cipher Text Length");
    }

    else
    {
        temp=0;
        for(i=0;i<ctipl;i++)
        {
            ctipc[i]=ct.charAt(i);
            ctipi[i]=ctipc[i];
```

```
    }

    for(i=0;i<ctipl;i++)
    {
        if(ctipi[i]<48||ctipi[i]>57)
        {
            temp=1;
        }
    }

    if(temp==1)
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Cipher Text");
    }
}
}while(temp==1);

do
{
    temp=0;
    while(temp==0)
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter the Key (A Number In The Range Of 0 Through 7):");
        if(indent==1)System.out.print("\t\t\t");
        try
        {
            key=Integer.parseInt(bfrd.readLine());
            temp=1;
        }
        catch(NumberFormatException E1nfe)
        {
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Key Should Be A Number");
            temp=0;
        }
    }
}

if(key<0||key>7)
{
    temp=1;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Invalid Key");
}
else
{
    temp=0;
}
}while(temp==1);

ctl=ctipl/3;
```

```
for(i=0;i<ctl;i++)
{
    j=i*3;
    cti[i]=0;
    for(k=0;k<3;k++)
    {
        if(k==0)
        {
            l=100;
        }
        else if(k==1)
        {
            l=10;
        }
        else
        {
            l=1;
        }
        cti[i]=cti[i]+(ctipi[j+k]-48)*l;
    }
}

for(i=0;i<ctl;i++)
{
    temp=cti[i];
    for(j=7;j>=0;j--)
    {
        pow=1;
        for(k=1;k<=j;k++)
        {
            pow=pow*2;
        }

        if(pow<=temp)
        {
            ctbinary[i][7-j]=1;
            temp=temp-pow;
        }
        else
        {
            ctbinary[i][7-j]=0;
        }
    }
}

for(i=0;i<ctl;i++)
{
    for(j=0;j<=7;j++)
    {
        tempbinary[j]=ctbinary[i][j];
    }
    for(j=0;j<=7;j++)
    {
        ctbinary[i][j]=tempbinary[(j+8+key)%8];
    }
}
```

```
    }
}

for(i=0;i<ctl;i++)
{
    temp=0;
    pow=1;
    for(j=7;j>=0;j--)
    {
        temp=temp+pow*ctbinary[i][j];
        pow=pow*2;
    }
    pti[i]=temp;
}

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Plain Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<ctl;i++)
{
    ptc[i]=(char)pti[i];
    System.out.print(ptc[i]);
}
System.out.println("");
}

public static void BitwiseExclusiveOr_enc()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int pow;
    int pti;
    int keyl;
    int digits;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int keyi[];
    keyi = new int[10000];

    int ptbinary[][];
    ptbinary = new int[10000][8];

    int keybinary[][];
```

```
keybinary = new int[10000][8];

char ptc[];
ptc = new char[10000];

char keyc[];
keyc = new char[10000];

String pt;
String key;

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Plain Text (A String Of Length Greater Than 2):");
    if(indent==1)System.out.print("\t\t\t");
    pt=bfrd.readLine();
    ptl=pt.length();
    if(ptl<3)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Plain Text Length");
    }
    else
    {
        temp=0;
    }
} while(temp==1);

for(i=0;i<ptl;i++)
{
    ptc[i]=pt.charAt(i);
    pti[i]=ptc[i];
}

for(i=0;i<ptl;i++)
{
    temp=pti[i];
    for(j=7;j>=0;j--)
    {
        pow=1;
        for(k=1;k<=j;k++)
        {
            pow=pow*2;
        }

        if(pow<=temp)
        {
            ptbinary[i][7-j]=1;
            temp=temp-pow;
        }
    }
}
```

```
        else
        {
            ptbinary[i][7-j]=0;
        }
    }
}

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Keyword (A String Of Length Greater Than 1):");
    if(indent==1)System.out.print("\t\t\t");
    key=bfrd.readLine();
    keyl=key.length();
    if(keyl<2)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Key Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<keyl;i++)
{
    keyc[i]=key.charAt(i);
    keyi[i]=keyc[i];
}

for(i=0;i<keyl;i++)
{
    temp=keyi[i];
    for(j=7;j>=0;j--)
    {
        pow=1;
        for(k=1;k<=j;k++)
        {
            pow=pow*2;
        }

        if(pow<=temp)
        {
            keybinary[i][7-j]=1;
            temp=temp-pow;
        }
        else
        {
            keybinary[i][7-j]=0;
        }
    }
}
```

```

    }
}

for(i=0;i<ptl;i++)
{
    keyi[i]=keyi[i%keyl];
    for(j=0;j<=7;j++)
    {
        keybinary[i][j]=keybinary[i%keyl][j];
    }
}

for(i=0;i<ptl;i++)
{
    for(j=0;j<=7;j++)
    {
        if(ptbinary[i][j]==0&&keybinary[i][j]==0)
        {
            ptbinary[i][j]=0;
        }
        else if(ptbinary[i][j]==0&&keybinary[i][j]==1)
        {
            ptbinary[i][j]=1;
        }
        else if(ptbinary[i][j]==1&&keybinary[i][j]==0)
        {
            ptbinary[i][j]=1;
        }
        else if(ptbinary[i][j]==1&&keybinary[i][j]==1)
        {
            ptbinary[i][j]=0;
        }
    }
}

for(i=0;i<ptl;i++)
{
    temp=0;
    pow=1;
    for(j=7;j>=0;j--)
    {
        temp=temp+pow*ptbinary[i][j];
        pow=pow*2;
    }
    cti[i]=temp;
}

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Cipher Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<ptl;i++)
{

```

```
temp=cti[i];
digits=0;
do
{
    digits=digits+1;
    temp=temp/10;
}while(temp!=0);
if(digits==0)
{
    System.out.print("000");
}
else if(digits==1)
{
    System.out.print("00");
}
else if(digits==2)
{
    System.out.print("0");
}
else if(digits==3)
{
    System.out.print("");
}
System.out.print(cti[i]);
}
System.out.println("");
}
```

```
public static void BitwiseExclusiveOr_dec()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int l;
    int pow;
    int cti;
    int ctipl;
    int keyl;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int ctipi[];
    ctipi = new int[30000];

    int keyi[];
    keyi = new int[10000];

    int ctbinary[][];
```



```
ctbinary = new int[10000][8];

int keybinary[][];
keybinary = new int[10000][8];

char ptc[];
ptc = new char[10000];

char ctc[];
ctc = new char[10000];

char ctipc[];
ctipc = new char[30000];

char keyc[];
keyc = new char[10000];

String ct;
String key;

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Cipher Text (A Sequence Of Numbers Of Length Greater Than 6 And
Of Length A Multiple Of 3):");
    if(indent==1)System.out.print("\t\t\t");
    ct=bfrd.readLine();
    ctipl=ct.length();

    if(ctipl<7||(ctipl%3)!=0)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Cipher Text Length");
    }

    else
    {
        temp=0;
        for(i=0;i<ctipl;i++)
        {
            ctipc[i]=ct.charAt(i);
            ctipi[i]=ctipc[i];
        }

        for(i=0;i<ctipl;i++)
        {
            if(ctipi[i]<48||ctipi[i]>57)
            {
                temp=1;
            }
        }
    }
}
```

```
        if(temp==1)
        {
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Invalid Cipher Text");
        }
    }
}while(temp==1);

ctl=ctipl/3;

for(i=0;i<ctl;i++)
{
    j=i*3;
    cti[i]=0;
    for(k=0;k<3;k++)
    {
        if(k==0)
        {
            l=100;
        }
        else if(k==1)
        {
            l=10;
        }
        else
        {
            l=1;
        }
        cti[i]=cti[i]+(ctipi[j+k]-48)*l;
    }
}

for(i=0;i<ctl;i++)
{
    temp=cti[i];
    for(j=7;j>=0;j--)
    {
        pow=1;
        for(k=1;k<=j;k++)
        {
            pow=pow*2;
        }

        if(pow<=temp)
        {
            ctbinary[i][7-j]=1;
            temp=temp-pow;
        }
        else
        {
            ctbinary[i][7-j]=0;
        }
    }
}
```

```
temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Keyword (A String Of Length Greater Than 1):");
    if(indent==1)System.out.print("\t\t\t");
    key=bfrd.readLine();
    keyl=key.length();
    if(keyl<2)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Key Length");
    }
    else
    {
        temp=0;
    }
} while(temp==1);

for(i=0;i<keyl;i++)
{
    keyc[i]=key.charAt(i);
    keyi[i]=keyc[i];
}

for(i=0;i<keyl;i++)
{
    temp=keyi[i];
    for(j=7;j>=0;j--)
    {
        pow=1;
        for(k=1;k<=j;k++)
        {
            pow=pow*2;
        }

        if(pow<=temp)
        {
            keybinary[i][7-j]=1;
            temp=temp-pow;
        }
        else
        {
            keybinary[i][7-j]=0;
        }
    }
}

for(i=0;i<ctl;i++)
{
    keyi[i]=keyi[i%keyl];
}
```

```
        for(j=0;j<=7;j++)
        {
            keybinary[i][j]=keybinary[i%keyl][j];
        }
    }

    for(i=0;i<ctl;i++)
    {
        for(j=0;j<=7;j++)
        {
            if(ctbinary[i][j]==0&&keybinary[i][j]==0)
            {
                ctbinary[i][j]=0;
            }
            else if(ctbinary[i][j]==0&&keybinary[i][j]==1)
            {
                ctbinary[i][j]=1;
            }
            else if(ctbinary[i][j]==1&&keybinary[i][j]==0)
            {
                ctbinary[i][j]=1;
            }
            else if(ctbinary[i][j]==1&&keybinary[i][j]==1)
            {
                ctbinary[i][j]=0;
            }
        }
    }

    for(i=0;i<ctl;i++)
    {
        temp=0;
        pow=1;
        for(j=7;j>=0;j--)
        {
            temp=temp+pow*ctbinary[i][j];
            pow=pow*2;
        }
        pti[i]=temp;
    }

    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("The Plain Text Is:");
    if(indent==1)System.out.print("\t\t\t");
    System.out.print("");
    for(i=0;i<ctl;i++)
    {
        ptc[i]=(char)pti[i];
        System.out.print(ptc[i]);
    }
    System.out.println("");
}
```

```
public static void BitwiseOneTimePad_enc()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int pow;
    int ptl;
    int keyl;
    int digits;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int keyi[];
    keyi = new int[10000];

    int ptbinary[][];
    ptbinary = new int[10000][8];

    int keybinary[][];
    keybinary = new int[10000][8];

    char ptc[];
    ptc = new char[10000];

    char keyc[];
    keyc = new char[10000];

    String pt;
    String key;

    temp=0;
    do
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter The Plain Text (A String Of Length Greater Than 2):");
        if(indent==1)System.out.print("\t\t\t");
        pt=bfrd.readLine();
        ptl=pt.length();
        if(ptl<3)
        {
            temp=1;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Invalid Plain Text Length");
        }
        else
        {
```

```
        temp=0;
    }
} while(temp==1);

for(i=0;i<ptl;i++)
{
    ptc[i]=pt.charAt(i);
    pti[i]=ptc[i];
}

for(i=0;i<ptl;i++)
{
    temp=pti[i];
    for(j=7;j>=0;j--)
    {
        pow=1;
        for(k=1;k<=j;k++)
        {
            pow=pow*2;
        }

        if(pow<=temp)
        {
            ptbinary[i][7-j]=1;
            temp=temp-pow;
        }
        else
        {
            ptbinary[i][7-j]=0;
        }
    }
}

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Keyword (A String Of Length Atleast As Long As Plain Text):");
    if(indent==1)System.out.print("\t\t\t");
    key=bfrd.readLine();
    keyl=key.length();
    if(keyl<ptl)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Key Length");
    }
    else
    {
        temp=0;
    }
} while(temp==1);
```

```
for(i=0;i<keyl;i++)
{
    keyc[i]=key.charAt(i);
    keyi[i]=keyc[i];
}

for(i=0;i<keyl;i++)
{
    temp=keyi[i];
    for(j=7;j>=0;j--)
    {
        pow=1;
        for(k=1;k<=j;k++)
        {
            pow=pow*2;
        }

        if(pow<=temp)
        {
            keybinary[i][7-j]=1;
            temp=temp-pow;
        }
        else
        {
            keybinary[i][7-j]=0;
        }
    }
}

for(i=0;i<ptl;i++)
{
    keyi[i]=keyi[i%keyl];
    for(j=0;j<=7;j++)
    {
        keybinary[i][j]=keybinary[i%keyl][j];
    }
}

for(i=0;i<ptl;i++)
{
    for(j=0;j<=7;j++)
    {
        if(ptbinary[i][j]==0&&keybinary[i][j]==0)
        {
            ptbinary[i][j]=0;
        }
        else if(ptbinary[i][j]==0&&keybinary[i][j]==1)
        {
            ptbinary[i][j]=1;
        }
        else if(ptbinary[i][j]==1&&keybinary[i][j]==0)
        {
            ptbinary[i][j]=1;
        }
        else if(ptbinary[i][j]==1&&keybinary[i][j]==1)
    }
```

```
        {
            ptbinary[i][j]=0;
        }
    }
}

for(i=0;i<ptl;i++)
{
    temp=0;
    pow=1;
    for(j=7;j>=0;j--)
    {
        temp=temp+pow*ptbinary[i][j];
        pow=pow*2;
    }
    cti[i]=temp;
}

if(indent==1)System.out.print("\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t");
System.out.println("The Cipher Text Is:");
if(indent==1)System.out.print("\t\t");
System.out.print("");
for(i=0;i<ptl;i++)
{
    temp=cti[i];
    digits=0;
    do
    {
        digits=digits+1;
        temp=temp/10;
    }while(temp!=0);
    if(digits==0)
    {
        System.out.print("000");
    }
    else if(digits==1)
    {
        System.out.print("00");
    }
    else if(digits==2)
    {
        System.out.print("0");
    }
    else if(digits==3)
    {
        System.out.print("");
    }
    System.out.print(cti[i]);
}
System.out.println("");
}

public static void BitwiseOneTimePad_dec()throws IOException
```



```
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int l;
    int pow;
    int ctl;
    int ctipl;
    int keyl;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int ctipi[];
    ctipi = new int[30000];

    int keyi[];
    keyi = new int[10000];

    int ctbinary[][];
    ctbinary = new int[10000][8];

    int keybinary[][];
    keybinary = new int[10000][8];

    char ptc[];
    ptc = new char[10000];

    char ctc[];
    ctc = new char[10000];

    char ctipc[];
    ctipc = new char[30000];

    char keyc[];
    keyc = new char[10000];

    String ct;
    String key;

    temp=0;
    do
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter The Cipher Text (A Sequence Of Numbers Of Length Greater Than 6 And
Of Length A Multiple Of 3):");
        if(indent==1)System.out.print("\t\t\t");
```

```
ct=bfrd.readLine();
ctipl=ct.length();

if(ctipl<7||(ctipl%3)!=0)
{
    temp=1;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Invalid Cipher Text Length");
}

else
{
    temp=0;
    for(i=0;i<ctipl;i++)
    {
        ctipc[i]=ct.charAt(i);
        ctipi[i]=ctipc[i];
    }

    for(i=0;i<ctipl;i++)
    {
        if(ctipi[i]<48||ctipi[i]>57)
        {
            temp=1;
        }
    }

    if(temp==1)
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Cipher Text");
    }
}
}while(temp==1);

ctl=ctipl/3;

for(i=0;i<ctl;i++)
{
    j=i*3;
    cti[i]=0;
    for(k=0;k<3;k++)
    {
        if(k==0)
        {
            l=100;
        }
        else if(k==1)
        {
            l=10;
        }
        else
        {
            l=1;
        }
    }
}
```

```

        cti[i]=cti[i]+(ctipi[j+k]-48)*1;
    }
}

for(i=0;i<ctl;i++)
{
    temp=cti[i];
    for(j=7;j>=0;j--)
    {
        pow=1;
        for(k=1;k<=j;k++)
        {
            pow=pow*2;
        }

        if(pow<=temp)
        {
            ctbinary[i][7-j]=1;
            temp=temp-pow;
        }
        else
        {
            ctbinary[i][7-j]=0;
        }
    }
}

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Keyword (A String Of Length Atleast As Long As One Third Of
Cipher Text):");
    if(indent==1)System.out.print("\t\t\t");
    key=bfrd.readLine();
    keyl=key.length();
    if(keyl<ctl)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Key Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<keyl;i++)
{
    keyc[i]=key.charAt(i);
    keyi[i]=keyc[i];
}

```

```
for(i=0;i<keyl;i++)
{
    temp=keyi[i];
    for(j=7;j>=0;j--)
    {
        pow=1;
        for(k=1;k<=j;k++)
        {
            pow=pow*2;
        }

        if(pow<=temp)
        {
            keybinary[i][7-j]=1;
            temp=temp-pow;
        }
        else
        {
            keybinary[i][7-j]=0;
        }
    }
}

for(i=0;i<ctl;i++)
{
    keyi[i]=keyi[i%keyl];
    for(j=0;j<=7;j++)
    {
        keybinary[i][j]=keybinary[i%keyl][j];
    }
}

for(i=0;i<ctl;i++)
{
    for(j=0;j<=7;j++)
    {
        if(ctbinary[i][j]==0&&keybinary[i][j]==0)
        {
            ctbinary[i][j]=0;
        }
        else if(ctbinary[i][j]==0&&keybinary[i][j]==1)
        {
            ctbinary[i][j]=1;
        }
        else if(ctbinary[i][j]==1&&keybinary[i][j]==0)
        {
            ctbinary[i][j]=1;
        }
        else if(ctbinary[i][j]==1&&keybinary[i][j]==1)
        {
            ctbinary[i][j]=0;
        }
    }
}
```

```
for(i=0;i<ctl;i++)
{
    temp=0;
    pow=1;
    for(j=7;j>=0;j--)
    {
        temp=temp+pow*ctbinary[i][j];
        pow=pow*2;
    }
    pti[i]=temp;
}

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Plain Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<ctl;i++)
{
    ptc[i]=(char)pti[i];
    System.out.print(ptc[i]);
}
System.out.println("");
}

public static void SquareTransposition_enc()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int l;
    int pti;
    int sq1;
    int sq2;
    int test;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    char ptc[];
    ptc = new char[10000];
    char ctc[];
    ctc = new char[10000];

    String pt;

    k=1;
```

```
test=1;

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Plain Text (A String Of Length Greater Than 2):");
    if(indent==1)System.out.print("\t\t\t");
    pt=bfrd.readLine();
    ptl=pt.length();
    if(ptl<3)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Plain Text Length");
    }
    else
    {
        temp=0;
    }
} while(temp==1);

for(i=0;i<ptl;i++)
{
    ptc[i]=pt.charAt(i);
    pti[i]=ptc[i];
}

i=0;

while(test==1)
{
    j=i+1;
    sq1=i*i;
    sq2=j*j;
    if(ptl>sq1&&ptl<=sq2)
    {
        if(ptl<sq2)
        {
            for(k=ptl;k<sq2;k++)
            {
                pti[k]=' ';
            }
            ptl=sq2;
        }
        k=j;
        test=0;
    }
    i++;
}

i=0;
l=0;
```

```
for(i=0;i<k;i++)
{
    for(j=0;j<k;j++)
    {
        temp=i+k*j;
        cti[l]=pti[temp];
        l=l+1;
    }
}

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Cipher Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<ptl;i++)
{
    ctc[i]=(char)cti[i];
    System.out.print(ctc[i]);
}
System.out.println("");
}

public static void SquareTransposition_dec()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int l;
    int ctl;
    int sq1;
    int sq2;
    int test;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    char ptc[];
    ptc = new char[10000];
    char ctc[];
    ctc = new char[10000];

    String ct;

    k=1;
    test=1;
```

```
temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Cipher Text (A String Of Length Greater Than 2):");
    if(indent==1)System.out.print("\t\t\t");
    ct=bfrd.readLine();
    ctl=ct.length();
    if(ctl<3)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Plain Text Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<ctl;i++)
{
    ctc[i]=ct.charAt(i);
    cti[i]=ctc[i];
}

i=0;

while(test==1)
{
    j=i+1;
    sq1=i*i;
    sq2=j*j;
    if(ctl>sq1&&ctl<=sq2)
    {
        if(ctl<sq2)
        {
            for(k=ctl;k<sq2;k++)
            {
                cti[k]=' ';
            }
            ctl=sq2;
        }
        k=j;
        test=0;
    }
    i++;
}

i=0;
l=0;

for(i=0;i<k;i++)
```



```
{
    for(j=0;j<k;j++)
    {
        temp=i+k*j;
        pti[l]=cti[temp];
        l=l+1;
    }
}

temp=0;

for(i=0;i<ctl;i++)
{
    if(pti[i]!=32)
    {
        temp=i;
    }
}

if(indent==1)System.out.print("\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t");
System.out.println("The Plain Text Is:");
if(indent==1)System.out.print("\t\t");
System.out.print("");
for(i=0;i<=temp;i++)
{
    ptc[i]=(char)pti[i];
    System.out.print(ptc[i]);
}
System.out.println("");
}

public static void ColumnarTransposition_enc()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int ptl;
    int keyl;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int keyi[];
    keyi = new int[10000];

    int keyiasc[][];
    keyiasc = new int[2][10000];
```

```
int cta[][];  
cta = new int[100][1000];  
  
char ptc[];  
ptc = new char[10000];  
  
char ctc[];  
ctc = new char[10000];  
  
char keyc[];  
keyc = new char[10000];  
  
String pt;  
String key;  
  
temp=0;  
do  
{  
    if(indent==1)System.out.print("\t\t\t");  
    System.out.println();  
    if(indent==1)System.out.print("\t\t\t");  
    System.out.println("Enter The Plain Text (A String Of Length Greater Than 2):");  
    if(indent==1)System.out.print("\t\t\t");  
    pt=bfrd.readLine();  
    ptl=pt.length();  
    if(ptl<3)  
    {  
        temp=1;  
        if(indent==1)System.out.print("\t\t\t");  
        System.out.println("Invalid Plain Text Length");  
    }  
    else  
    {  
        temp=0;  
    }  
}while(temp==1);  
  
for(i=0;i<ptl;i++)  
{  
    ptc[i]=pt.charAt(i);  
    pti[i]=ptc[i];  
}  
  
temp=0;  
do  
{  
    if(indent==1)System.out.print("\t\t\t");  
    System.out.println();  
    if(indent==1)System.out.print("\t\t\t");  
    System.out.println("Enter The Keyword (A String Of Length Greater Than 1 And Less Than 100  
And Less Than Length Of Plain Text):");  
    if(indent==1)System.out.print("\t\t\t");  
    key=bfrd.readLine();  
    keyl=key.length();
```

```
if(keyl<2||keyl>=100||keyl>=ptl)
{
    temp=1;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Invalid Key Length");
}
else
{
    temp=0;
}
}while(temp==1);

for(i=0;i<keyl;i++)
{
    keyc[i]=key.charAt(i);
    keyi[i]=keyc[i];
    keyiasc[0][i]=i;
    keyiasc[1][i]=keyi[i];
}

for(i=0;i<keyl;i++)
{
    for(j=i;j<keyl;j++)
    {
        if(keyiasc[1][j]<keyiasc[1][i])
        {
            temp=keyiasc[1][j];
            keyiasc[1][j]=keyiasc[1][i];
            keyiasc[1][i]=temp;

            temp=keyiasc[0][j];
            keyiasc[0][j]=keyiasc[0][i];
            keyiasc[0][i]=temp;
        }
    }
}

temp=0;
i=ptl;
while(ptl%keyl!=0)
{
    pti[i]=' ';
    i=i+1;
    ptl=ptl+1;
}

k=0;
for(i=0;i<ptl/keyl;i++)
{
    for(j=0;j<keyl;j++)
    {
        cta[i][j]=pti[k];
        k=k+1;
    }
}
```

```
k=0;
for(i=0;i<keyl;i++)
{
    for(j=0;j<ptl/keyl;j++)
    {
        cti[k]=cta[j][keyiasc[0][i]];
        k=k+1;
    }
}

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Cipher Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<ptl;i++)
{
    ctc[i]=(char)cti[i];
    System.out.print(ctc[i]);
}
System.out.println("");
}

public static void ColumnarTransposition_dec()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int ctl;
    int keyl;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int keyi[];
    keyi = new int[10000];

    int keyiasc[][];
    keyiasc = new int[2][10000];

    int cta[][];
    cta = new int[100][1000];

    char ptc[];
    ptc = new char[10000];

    char ctc[];
```

```
ctc = new char[10000];

char keyc[];
keyc = new char[10000];

String ct;
String key;

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Cipher Text (A String Of Length Greater Than 2):");
    if(indent==1)System.out.print("\t\t\t");
    ct=bfrd.readLine();
    ctl=ct.length();
    if(ctl<3)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Cipher Text Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<ctl;i++)
{
    ctc[i]=ct.charAt(i);
    cti[i]=ctc[i];
}

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Keyword (A String Of Length Greater Than 1 And Less Than 100
And Less Than Length Of Cipher Text):");
    if(indent==1)System.out.print("\t\t\t");
    key=bfrd.readLine();
    keyl=key.length();
    if(keyl<2||keyl>=100||keyl>=ctl)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Key Length");
    }
    else
    {
```

```
        temp=0;
    }
} while(temp==1);

for(i=0;i<keyl;i++)
{
    keyc[i]=key.charAt(i);
    keyi[i]=keyc[i];
    keyiasc[0][i]=i;
    keyiasc[1][i]=keyi[i];
}

for(i=0;i<keyl;i++)
{
    for(j=i;j<keyl;j++)
    {
        if(keyiasc[1][j]<keyiasc[1][i])
        {
            temp=keyiasc[1][j];
            keyiasc[1][j]=keyiasc[1][i];
            keyiasc[1][i]=temp;

            temp=keyiasc[0][j];
            keyiasc[0][j]=keyiasc[0][i];
            keyiasc[0][i]=temp;
        }
    }
}

temp=0;
i=ctl;
while(ctl%keyl!=0)
{
    cti[i]=' ';
    i=i+1;
    ctl=ctl+1;
}

k=0;
for(i=0;i<keyl;i++)
{
    for(j=0;j<ctl/keyl;j++)
    {
        cta[j][keyiasc[0][i]]=cti[k];
        k++;
    }
}

k=0;
for(i=0;i<ctl/keyl;i++)
{
    for(j=0;j<keyl;j++)
    {
        pti[k]=cta[i][j];
        k++;
    }
}
```

```
    }
}

temp=0;
for(i=0;i<ctl;i++)
{
    if(pti[i]!=' ')
    {
        temp=i;
    }
}
ctl=temp+1;

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Plain Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<ctl;i++)
{
    ptc[i]=(char)pti[i];
    System.out.print(ptc[i]);
}
System.out.println("");
}

public static void DoubleTransposition_enc()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int pti;
    int keyl;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    int keyi[];
    keyi = new int[10000];

    int keyiasc[][];
    keyiasc = new int[2][10000];

    int cta[][];
    cta = new int[100][1000];

    char ptc[];
    ptc = new char[10000];
```

```
char ctc[];
ctc = new char[10000];

char keyc[];
keyc = new char[10000];

String pt;
String key;

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Plain Text (A String Of Length Greater Than 2):");
    if(indent==1)System.out.print("\t\t\t");
    pt=bfrd.readLine();
    ptl=pt.length();
    if(ptl<3)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Plain Text Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<ptl;i++)
{
    ptc[i]=pt.charAt(i);
    pti[i]=ptc[i];
}

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The First Keyword (A String Of Length Greater Than 1 And Less Than 100 And Less Than Length Of Plain Text):");
    if(indent==1)System.out.print("\t\t\t");
    key=bfrd.readLine();
    keyl=key.length();
    if(keyl<2||keyl>=100||keyl>=ptl)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Key Length");
    }
}
```



```
        else
        {
            temp=0;
        }
    }while(temp==1);

    for(i=0;i<keyl;i++)
    {
        keyc[i]=key.charAt(i);
        keyi[i]=keyc[i];
        keyiasc[0][i]=i;
        keyiasc[1][i]=keyi[i];
    }

    for(i=0;i<keyl;i++)
    {
        for(j=i;j<keyl;j++)
        {
            if(keyiasc[1][j]<keyiasc[1][i])
            {
                temp=keyiasc[1][j];
                keyiasc[1][j]=keyiasc[1][i];
                keyiasc[1][i]=temp;

                temp=keyiasc[0][j];
                keyiasc[0][j]=keyiasc[0][i];
                keyiasc[0][i]=temp;
            }
        }
    }

    temp=0;
    i=ptl;
    while(ptl%keyl!=0)
    {
        pti[i]=' ';
        i=i+1;
        ptl=ptl+1;
    }

    k=0;
    for(i=0;i<ptl/keyl;i++)
    {
        for(j=0;j<keyl;j++)
        {
            cta[i][j]=pti[k];
            k=k+1;
        }
    }

    k=0;
    for(i=0;i<keyl;i++)
    {
        for(j=0;j<ptl/keyl;j++)
        {
```

```

        cti[k]=cta[j][keyiasc[0][i]];
        k=k+1;
    }
}

for(i=0;i<ptl;i++)
{
    ctc[i]=(char)cti[i];
    ptc[i]=ctc[i];
    pti[i]=ptc[i];
}

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Second Keyword (A String Of Length Greater Than 1 And Less Than
100 And Less Than Length Of Plain Text):");
    if(indent==1)System.out.print("\t\t\t");
    key=bfrd.readLine();
    keyl=key.length();
    if(keyl<2||keyl>=100||keyl>=ptl)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Key Length");
    }
    else
    {
        temp=0;
    }
} while(temp==1);

for(i=0;i<keyl;i++)
{
    keyc[i]=key.charAt(i);
    keyi[i]=keyc[i];
    keyiasc[0][i]=i;
    keyiasc[1][i]=keyi[i];
}

for(i=0;i<keyl;i++)
{
    for(j=i;j<keyl;j++)
    {
        if(keyiasc[1][j]<keyiasc[1][i])
        {
            temp=keyiasc[1][j];
            keyiasc[1][j]=keyiasc[1][i];
            keyiasc[1][i]=temp;

            temp=keyiasc[0][j];
            keyiasc[0][j]=keyiasc[0][i];

```

```

        keyiasc[0][i]=temp;
    }
}

temp=0;
i=ptl;
while(ptl%keyl!=0)
{
    pti[i]=' ';
    i=i+1;
    ptl=ptl+1;
}

k=0;
for(i=0;i<ptl/keyl;i++)
{
    for(j=0;j<keyl;j++)
    {
        cta[i][j]=pti[k];
        k=k+1;
    }
}

k=0;
for(i=0;i<keyl;i++)
{
    for(j=0;j<ptl/keyl;j++)
    {
        cti[k]=cta[j][keyiasc[0][i]];
        k=k+1;
    }
}

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Cipher Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<ptl;i++)
{
    ctc[i]=(char)cti[i];
    System.out.print(ctc[i]);
}
System.out.println("");
}

public static void DoubleTransposition_dec()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;

```

```
int ctl;
int keyl;
int temp;

int pti[];
pti = new int[10000];

int cti[];
cti = new int[10000];

int keyi[];
keyi = new int[10000];

int keyiasc[][];
keyiasc = new int[2][10000];

int cta[][];
cta = new int[100][1000];

char ptc[];
ptc = new char[10000];

char ctc[];
ctc = new char[10000];

char keyc[];
keyc = new char[10000];

String ct;
String key;

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Cipher Text (A String Of Length Greater Than 2):");
    if(indent==1)System.out.print("\t\t\t");
    ct=bfrd.readLine();
    ctl=ct.length();
    if(ctl<3)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Cipher Text Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<ctl;i++)
{
```

```

        ctc[i]=ct.charAt(i);
        cti[i]=ctc[i];
    }

    temp=0;
    do
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter The Second Keyword (A String Of Length Greater Than 1 And Less Than
100 And Less Than Length Of Cipher Text):");
        if(indent==1)System.out.print("\t\t\t");
        key=bfrd.readLine();
        keyl=key.length();
        if(keyl<2||keyl>=100||keyl>=ctl)
        {
            temp=1;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Invalid Key Length");
        }
        else
        {
            temp=0;
        }
    }while(temp==1);

    for(i=0;i<keyl;i++)
    {
        keyc[i]=key.charAt(i);
        keyi[i]=keyc[i];
        keyiasc[0][i]=i;
        keyiasc[1][i]=keyi[i];
    }

    for(i=0;i<keyl;i++)
    {
        for(j=i;j<keyl;j++)
        {
            if(keyiasc[1][j]<keyiasc[1][i])
            {
                temp=keyiasc[1][j];
                keyiasc[1][j]=keyiasc[1][i];
                keyiasc[1][i]=temp;

                temp=keyiasc[0][j];
                keyiasc[0][j]=keyiasc[0][i];
                keyiasc[0][i]=temp;
            }
        }
    }

    temp=0;
    i=ctl;
    while(ctl%keyl!=0)

```

```

    {
        cti[i]=' ';
        i=i+1;
        ctl=ctl+1;
    }

    k=0;
    for(i=0;i<keyl;i++)
    {
        for(j=0;j<ctl/keyl;j++)
        {
            cta[j][keyiasc[0][i]]=cti[k];
            k++;
        }
    }

    k=0;
    for(i=0;i<ctl/keyl;i++)
    {
        for(j=0;j<keyl;j++)
        {
            pti[k]=cta[i][j];
            k++;
        }
    }

    temp=0;
    for(i=0;i<ctl;i++)
    {
        if(pti[i]!=' ')
        {
            temp=i;
        }
    }
    ctl=temp+1;

    for(i=0;i<ctl;i++)
    {
        ptc[i]=(char)pti[i];
        ctc[i]=ptc[i];
        cti[i]=ctc[i];
    }

    temp=0;
    do
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter The First Keyword (A String Of Length Greater Than 1 And Less Than
100 And Less Than Length Of Cipher Text):");
        if(indent==1)System.out.print("\t\t\t");
        key=bfrd.readLine();
        keyl=key.length();
        if(keyl<2||keyl>=100||keyl>=ctl)

```

```
{
    temp=1;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Invalid Key Length");
}
else
{
    temp=0;
}
}while(temp==1);

while(ctl%keyl!=0)
{
    cti[ctl]=' ';
    ctl++;
}

for(i=0;i<keyl;i++)
{
    keyc[i]=key.charAt(i);
    keyi[i]=keyc[i];
    keyiasc[0][i]=i;
    keyiasc[1][i]=keyi[i];
}

for(i=0;i<keyl;i++)
{
    for(j=i;j<keyl;j++)
    {
        if(keyiasc[1][j]<keyiasc[1][i])
        {
            temp=keyiasc[1][j];
            keyiasc[1][j]=keyiasc[1][i];
            keyiasc[1][i]=temp;

            temp=keyiasc[0][j];
            keyiasc[0][j]=keyiasc[0][i];
            keyiasc[0][i]=temp;
        }
    }
}

temp=0;
i=ctl;
while(ctl%keyl!=0)
{
    cti[i]=' ';
    i=i+1;
    ctl=ctl+1;
}

k=0;
for(i=0;i<keyl;i++)
{
    for(j=0;j<ctl/keyl;j++)
```

```

        {
            cta[j][keyiasc[0][i]]=cti[k];
            k++;
        }
    }

    k=0;
    for(i=0;i<ctl/keyl;i++)
    {
        for(j=0;j<keyl;j++)
        {
            pti[k]=cta[i][j];
            k++;
        }
    }

    temp=0;
    for(i=0;i<ctl;i++)
    {
        if(pti[i]!=' ')
        {
            temp=i;
        }
    }
    ctl=temp+1;

    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("The Plain Text Is:");
    if(indent==1)System.out.print("\t\t\t");
    System.out.print("");
    for(i=0;i<ctl;i++)
    {
        ptc[i]=(char)pti[i];
        System.out.print(ptc[i]);
    }
    System.out.println("");
}

public static void MyszkowskiTransposition_enc()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int l;
    int ptl;
    int keyl;
    int temp;

    int pti[];
    pti = new int[10000];

```



```
int cti[];
cti = new int[10000];

int keyi[];
keyi = new int[10000];

int keyiasc[][];
keyiasc = new int[2][10000];

int cta[][];
cta = new int[100][1000];

char ptc[];
ptc = new char[10000];

char ctc[];
ctc = new char[10000];

char keyc[];
keyc = new char[10000];

String pt;
String key;

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Plain Text (A String Of Length Greater Than 2):");
    if(indent==1)System.out.print("\t\t\t");
    pt=bfrd.readLine();
    ptl=pt.length();
    if(ptl<3)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Plain Text Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<ptl;i++)
{
    ptc[i]=pt.charAt(i);
    pti[i]=ptc[i];
}

temp=0;
do
{
```

```
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter The Keyword (A String Of Length Greater Than 1 And Less Than 100
And Less Than Length Of Plain Text):");
        if(indent==1)System.out.print("\t\t\t");
        key=bfrd.readLine();
        keyl=key.length();
        if(keyl<2||keyl>=100||keyl>=ptl)
        {
            temp=1;
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Invalid Key Length");
        }
        else
        {
            temp=0;
        }
    }while(temp==1);

    for(i=0;i<keyl;i++)
    {
        keyc[i]=key.charAt(i);
        keyi[i]=keyc[i];
        keyiasc[1][i]=keyi[i];
    }

    k=0;
    temp=0;
    for(i=0;i<=255;i++)
    {
        for(j=0;j<keyl;j++)
        {
            if(keyiasc[1][j]==i)
            {
                keyiasc[0][j]=k;
                temp=1;
            }
        }
        if(temp==1)
        {
            temp=0;
            k=k+1;
        }
    }

    temp=0;
    i=ptl;
    while(ptl%keyl!=0)
    {
        pti[i]=' ';
        i=i+1;
        ptl=ptl+1;
    }
```

```

temp=0;
for(i=0;i<ptl/keyl;i++)
{
    for(j=0;j<keyl;j++)
    {
        cta[i][j]=pti[temp];
        temp=temp+1;
    }
}

temp=k;
l=0;
for(i=0;i<temp;i++)
{
    for(j=0;j<ptl/keyl;j++)
    {
        for(k=0;k<keyl;k++)
        {
            if(keyiasc[0][k]==i)
            {
                cti[l]=cta[j][k];
                l=l+1;
            }
        }
    }
}

if(indent==1)System.out.print("\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t");
System.out.println("The Cipher Text Is:");
if(indent==1)System.out.print("\t\t");
System.out.print("");
for(i=0;i<ptl;i++)
{
    ctc[i]=(char)cti[i];
    System.out.print(ctc[i]);
}
System.out.println("");
}

public static void MyszkowskiTransposition_dec()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int l;
    int ctl;
    int keyl;
    int temp;

    int pti[];
    pti = new int[10000];

```

```
int cti[];
cti = new int[10000];

int keyi[];
keyi = new int[10000];

int keyiasc[][];
keyiasc = new int[2][10000];

int cta[][];
cta = new int[100][1000];

char ptc[];
ptc = new char[10000];

char ctc[];
ctc = new char[10000];

char keyc[];
keyc = new char[10000];

String ct;
String key;

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Cipher Text (A String Of Length Greater Than 2):");
    if(indent==1)System.out.print("\t\t\t");
    ct=bfrd.readLine();
    ctl=ct.length();
    if(ctl<3)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Cipher Text Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<ctl;i++)
{
    ctc[i]=ct.charAt(i);
    cti[i]=ctc[i];
}

temp=0;
do
```

```
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Keyword (A String Of Length Greater Than 1 And Less Than 100
And Less Than Length Of Cipher Text:");
    if(indent==1)System.out.print("\t\t\t");
    key=bfrd.readLine();
    keyl=key.length();
    if(keyl<2||keyl>=100||keyl>=ctl)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Key Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<keyl;i++)
{
    keyc[i]=key.charAt(i);
    keyi[i]=keyc[i];
    keyiasc[1][i]=keyi[i];
}

k=0;
temp=0;
for(i=0;i<=255;i++)
{
    for(j=0;j<keyl;j++)
    {
        if(keyiasc[1][j]==i)
        {
            keyiasc[0][j]=k;
            temp=1;
        }
    }
    if(temp==1)
    {
        temp=0;
        k=k+1;
    }
}

temp=0;
i=ctl;
while(ctl%keyl!=0)
{
    cti[i]=' ';
    i=i+1;
    ctl=ctl+1;
}
```

```

temp=k;
l=0;
for(i=0;i<temp;i++)
{
    for(j=0;j<ctl/keyl;j++)
    {
        for(k=0;k<keyl;k++)
        {
            if(keyiasc[0][k]==i)
            {
                cta[j][k]=cti[l];
                l=l+1;
            }
        }
    }
}

temp=0;
for(i=0;i<ctl/keyl;i++)
{
    for(j=0;j<keyl;j++)
    {
        pti[temp]=cta[i][j];
        temp=temp+1;
    }
}

temp=0;
for(i=0;i<ctl;i++)
{
    if(pti[i]!=' ')
    {
        temp=i;
    }
}
ctl=temp+1;

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Plain Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<ctl;i++)
{
    ptc[i]=(char)pti[i];
    System.out.print(ptc[i]);
}
System.out.println("");
}

public static void ChineseTransposition_enc()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

```

```
int i;
int j;
int k;
int l;
int pti;
int key;
int rows;
int cols;
int temp;

int pti[];
pti = new int[10000];

int cti[];
cti = new int[10000];

int cta[][];
cta = new int[100][10000];

char ptc[];
ptc = new char[10000];

char ctc[];
ctc = new char[10000];

String pt;

key=0;

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Plain Text (A String Of Length Greater Than 2):");
    if(indent==1)System.out.print("\t\t\t");
    pt=bfrd.readLine();
    pti=pt.length();
    if(pti<3)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Plain Text Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<pti;i++)
{
    ptc[i]=pt.charAt(i);
```

```
        pti[i]=ptc[i];
    }

    do
    {
        temp=0;
        while(temp==0)
        {
            if(indent==1)System.out.print("\t\t\t");
            System.out.println();
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Enter The Key (A Number Greater Than 1 And Less Than 100 And Less Than
Length Of Plain Text):");
            if(indent==1)System.out.print("\t\t\t");
            try
            {
                key=Integer.parseInt(bfrd.readLine());
                temp=1;
            }
            catch(NumberFormatException E1nfe)
            {
                if(indent==1)System.out.print("\t\t\t");
                System.out.println("Key Should Be A Number");
                temp=0;
            }
        }
    }

    if(key<2||key>99||key>=pt.length())
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Key");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

temp=0;
i=ptl;
while(ptl%key!=0)
{
    pti[i]=' ';
    i=i+1;
    ptl=ptl+1;
}

rows=key;
cols=ptl/key;

k=0;
l=0;
for(i=(cols-1);i>=0;i--)
{
```



```
        if(k%2==0)
        {
            for(j=0;j<rows;j++)
            {
                cta[j][i]=pti[l];
                l=l+1;
            }
        }
        else
        {
            for(j=(rows-1);j>=0;j--)
            {
                cta[j][i]=pti[l];
                l=l+1;
            }
        }
        k=k+1;
    }

    k=0;
    for(i=0;i<rows;i++)
    {
        for(j=0;j<cols;j++)
        {
            cti[k]=cta[i][j];
            k=k+1;
        }
    }

    if(indent==1)System.out.print("\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t");
    System.out.println("The Cipher Text Is:");
    if(indent==1)System.out.print("\t\t");
    System.out.print("");
    for(i=0;i<ptl;i++)
    {
        ctc[i]=(char)cti[i];
        System.out.print(ctc[i]);
    }
    System.out.println("");
}

public static void ChineseTransposition_dec()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int l;
    int ctl;
    int key;
    int rows;
    int cols;
```

```
int temp;

int pti[];
pti = new int[10000];

int cti[];
cti = new int[10000];

int cta[][];
cta = new int[100][10000];

char ptc[];
ptc = new char[10000];

char ctc[];
ctc = new char[10000];

String ct;

key=0;

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Cipher Text (A String Of Length Greater Than 2):");
    if(indent==1)System.out.print("\t\t\t");
    ct=bfrd.readLine();
    ctl=ct.length();
    if(ctl<3)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Cipher Text Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

for(i=0;i<ctl;i++)
{
    ctc[i]=ct.charAt(i);
    cti[i]=ctc[i];
}

do
{
    temp=0;
    while(temp==0)
    {
        if(indent==1)System.out.print("\t\t\t");
```

```
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter The Key (A Number Greater Than 1 And Less Than 100 And Less Than
Length Of Cipher Text):");
        if(indent==1)System.out.print("\t\t\t");
        try
        {
            key=Integer.parseInt(bfrd.readLine());
            temp=1;
        }
        catch(NumberFormatException E1nfe)
        {
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Key Should Be A Number");
            temp=0;
        }
    }

    if(key<2||key>99||key>=ct.length())
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Key");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

temp=0;
i=ctl;
while(ctl%key!=0)
{
    cti[i]=' ';
    i=i+1;
    ctl=ctl+1;
}

rows=key;
cols=ctl/key;

k=0;
for(i=0;i<rows;i++)
{
    for(j=0;j<cols;j++)
    {
        cta[i][j]=cti[k];
        k=k+1;
    }
}

k=0;
l=0;
for(i=(cols-1);i>=0;i--)
```

```
{
    if(k%2==0)
    {
        for(j=0;j<rows;j++)
        {
            pti[l]=cta[j][i];
            l=l+1;
        }
    }
    else
    {
        for(j=(rows-1);j>=0;j--)
        {
            pti[l]=cta[j][i];
            l=l+1;
        }
    }
    k=k+1;
}

temp=0;
for(i=0;i<ctl;i++)
{
    if(pti[i]!=32)
    {
        temp=i;
    }
}

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Plain Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<=temp;i++)
{
    ptc[i]=(char)pti[i];
    System.out.print(ptc[i]);
}
System.out.println("");
}

public static void RailFence_enc()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int l;
    int m;
    int n;
    int o;
    int p;
```

```
int q;
int r;
int s;
int k1;
int k2;
int key;
int ptl;
int sq1;
int sq2;
int temp;
int test;

int pti[];
pti = new int[10000];

int cti[];
cti = new int[10000];

char ptc[];
ptc = new char[10000];
char ctc[];
ctc = new char[10000];

String pt;

k=1;
m=1;
n=1;

key=0;

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Plain Text (A String Of Length Greater Than 2):");
    if(indent==1)System.out.print("\t\t\t");
    pt=bfrd.readLine();
    ptl=pt.length();
    if(ptl<3)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Plain Text Length");
    }
    else
    {
        temp=0;
    }
}while(temp==1);

do
{
```

```
temp=0;
while(temp==0)
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Key (A Number Greater Than 1 And Less Than The Length Of The
Plain Text):");
    if(indent==1)System.out.print("\t\t\t");
    try
    {
        key=Integer.parseInt(bfrd.readLine());
        temp=1;
    }
    catch(NumberFormatException E1nfe)
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Key Should Be A Number");
        temp=0;
    }
}

if(key<2||key>=pt.length())
{
    temp=1;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Invalid Key");
}
else
{
    temp=0;
}
}while(temp==1);

for(i=0;i<ptl;i++)
{
    ptc[i]=pt.charAt(i);
    pti[i]=ptc[i];
}

test=0;
i=0;
j=1;
l=key-2;

do
{
    k=key*j+i;
    if(ptl==k)
    {
        test=1;
        n=j;
    }
    i=l*j;
    j=j+1;
```

```
}while(k<=ptl);

i=0;
j=0;
while(test==0)
{
    k1=key*j+i;
    i=l*j;
    j=j+1;
    k2=key*j+i;
    if(ptl>k1&&ptl<k2)
    {
        for(k=ptl;k<k2;k++)
        {
            pti[k]=' ';
        }
        test=1;
        ptl=k2;
        n=j;
    }
}

for(k=0;k<key;k++)
{
    r=key-1;
    if(k==0)
    {
        for(j=0,i=1;j<n;j++,i=i+1)
        {
            m=key*j+i;
            s=m-1;
            cti[j]=pti[s];
        }
        m=n;
    }
    else if(k==r)
    {
        for(j=1,i=0;j<=n;j++,i=i+1)
        {
            o=key*j+i;
            s=o-1;
            cti[m]=pti[s];
            m=m+1;
        }
    }
    else
    {
        for(j=0,i=1;j<n;j++,i=i+1)
        {
            o=key*j+i;
            if(o==1)
            {
                p=o+k;
                s=p-1;
                cti[m]=pti[s];
            }
        }
    }
}
```

```

        m=m+1;
    }

    else
    {
        for(q=1;q<3;q++)
        {
            if(q==1)
            {
                p=o-k;
                s=p-1;
                cti[m]=pti[s];
                m=m+1;
            }
            else
            {
                p=o+k;
                s=p-1;
                cti[m]=pti[s];
                m=m+1;
            }
        }
    }
}

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Cipher Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
for(i=0;i<ptl;i++)
{
    ctc[i]=(char)cti[i];
    System.out.print(ctc[i]);
}
System.out.println("");
}

public static void RailFence_dec()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int j;
    int k;
    int l;
    int m;
    int n;
    int o;
    int p;
    int q;
    int r;

```



```
int s;
int t;
int u;
int v;
int w;
int x;
int y;
int k1;
int k2;
int key;
int ctl;
int ptl;
int temp;
int test;

int pti[];
pti = new int[10000];

int cti[];
cti = new int[10000];

char ptc[];
ptc = new char[10000];

char ctc[];
ctc = new char[10000];

String ct;

k=1;
m=1;
n=1;
o=1;

key=0;

temp=0;
do
{
    if(indent==1)System.out.print("\t\t\t");
    System.out.println();
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Enter The Cipher Text (A String Of Length Greater Than 2):");
    if(indent==1)System.out.print("\t\t\t");
    ct=bfrd.readLine();
    ctl=ct.length();
    if(ctl<3)
    {
        temp=1;
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Invalid Cipher Text Length");
    }
    else
    {
        temp=0;
    }
}
```

```

    }
}while(temp==1);

do
{
    temp=0;
    while(temp==0)
    {
        if(indent==1)System.out.print("\t\t\t");
        System.out.println();
        if(indent==1)System.out.print("\t\t\t");
        System.out.println("Enter The Key (A Number Greater Than 1 And Less Than The Length Of The
Cipher Text):");
        if(indent==1)System.out.print("\t\t\t");
        try
        {
            key=Integer.parseInt(bfrd.readLine());
            temp=1;
        }
        catch(NumberFormatException E1nfe)
        {
            if(indent==1)System.out.print("\t\t\t");
            System.out.println("Key Should Be A Number");
            temp=0;
        }
    }
}

if(key<2||key>=ct.length())
{
    temp=1;
    if(indent==1)System.out.print("\t\t\t");
    System.out.println("Invalid Key");
}
else
{
    temp=0;
}
}while(temp==1);

for(i=0;i<ctl;i++)
{
    ctc[i]=ct.charAt(i);
    cti[i]=ctc[i];
}

test=0;
i=0;
j=1;
l=key-2;

do
{
    k=key*j+i;
    if(ctl==k)
    {

```

```
        test=1;
        n=j;
    }
    i=l*j;
    j=j+1;
}while(k<=ctl);

i=0;
j=0;
while(test==0)
{
    k1=key*j+i;
    i=l*j;
    j=j+1;
    k2=key*j+i;
    if(ctl>k1&&ctl<k2)
    {
        for(k=ctl;k<k2;k++)
        {
            cti[k]=' ';
        }
        test=1;
        ctl=k2;
        n=j;
    }
}

if(key!=2)
{
    m=(2*n)-1;
}
else if(key==2)
{
    m=n;
}
p=(2*key)-1;

j=0;
for(k=0;k<n;k++)
{
    if(k==0)
    {
        for(i=1;i<=key;i++)
        {
            if(i==1)
            {
                o=1;
                w=o-1;
                pti[j]=cti[w];
                j=j+1;
            }
            else if(i==2)
            {
                o=o+n;
```

```
        w=o-1;
        pti[j]=cti[w];
        j=j+1;
    }
    else
    {
        o=o+m;
        w=o-1;
        pti[j]=cti[w];
        j=j+1;
    }
}
}

else
{
    for(i=1;i<p;i++)
    {
        q=k-1;
        r=m-k;
        s=n+q;
        t=n+k;
        u=(p-1)/2;
        v=(p+1)/2;
        x=p-1;
        y=m-1;
        if(i==u)
        {
            if(key==2)
            {
                o=o-y;
                w=o-1;
                pti[j]=cti[w];
                j=j+1;
            }
            else
            {
                o=o-s;
                w=o-1;
                pti[j]=cti[w];
                j=j+1;
            }
        }
        else if(i==v)
        {
            if(key==2)
            {
                o=o+n;
                w=o-1;
                pti[j]=cti[w];
                j=j+1;
            }
            else
            {
                o=o+t;
```

```
        w=o-1;
        pti[j]=cti[w];
        j=j+1;
    }
}
else if(i==1)
{
    o=o-r;
    w=o-1;
    pti[j]=cti[w];
    j=j+1;
}

else if(i==x)
{
    o=o+r;
    w=o-1;
    pti[j]=cti[w];
    j=j+1;
}

else if(i<u)
{
    o=o-m;
    w=o-1;
    pti[j]=cti[w];
    j=j+1;
}
else if(i>v)
{
    o=o+m;
    w=o-1;
    pti[j]=cti[w];
    j=j+1;
}
}
}

for(i=0;i<ctl;i++)
{
    if(pti[i]!=32)
    {
        j=i;
    }
}

ptl=j;

if(indent==1)System.out.print("\t\t\t");
System.out.println();
if(indent==1)System.out.print("\t\t\t");
System.out.println("The Plain Text Is:");
if(indent==1)System.out.print("\t\t\t");
System.out.print("");
```

```
    for(i=0;i<=ptl;i++)
    {
        ptc[i]=(char)pti[i];
        System.out.print(ptc[i]);
    }
    System.out.println("");
}
```

```
//***** End Of Program *****
*****//
```

# OUTPUT

HOME: IMPLEMENTATION OF CRYPTOGRAPHY IN JAVA

1. About
2. Ciphers
3. Author
4. Exit

Enter Choice: 1

HOME.1. ABOUT: IMPLEMENTATION OF CRYPTOGRAPHY IN JAVA

Cryptography is the practice and study of techniques for secure communication in the presence of third parties (called adversaries). This program demonstrates various ciphers in java. It accepts the Plain Text and Key (wherever applicable) from the user and generates the Cipher Text using various Encryption algorithms. The Plain Text is then obtained from the Cipher Text using the same Key by applying the respective Decryption algorithm.

HOME: IMPLEMENTATION OF CRYPTOGRAPHY IN JAVA

1. About
2. Ciphers
3. Author
4. Exit

Enter Choice: 3

HOME.3. AUTHOR

Name: Omkar S. Nath

Class: 10–D

Roll Number: 35

Computer Code: 230438

HOME: IMPLEMENTATION OF CRYPTOGRAPHY IN JAVA

1. About
2. Ciphers
3. Author
4. Exit

Enter Choice: 2

HOME.2. CIPHERS

1. Rotation Cipher
2. Atbash Cipher
3. Caesar Positive Shift Cipher
4. Caesar Negative Shift Cipher
5. One Time Pad Cipher
6. Monoalphabetic Substitution Cipher
7. Polyalphabetic Substitution Cipher
8. Autokey Cipher
9. Running Key Cipher
10. Polybius Square Cipher

11. Bifid Cipher
12. Trifid Cipher
13. Bitwise Rotation Cipher
14. Bitwise Atbash Cipher
15. Bitwise Binary Compliment Cipher
16. Bitwise Left Shift Cipher
17. Bitwise Right Shift Cipher
18. Bitwise Exclusive Or Cipher
19. Bitwise One Time Pad Cipher
20. Square Transposition Cipher
21. Columnar Transposition Cipher
22. Double Transposition Cipher
23. Myszkowski Transposition Cipher
24. Chinese Transposition Cipher
25. Rail Fence Cipher
26. Back To HOME

Enter Choice: 1

#### HOME.2.1. ROTATION CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 1

##### HOME.2.1.1. ABOUT: ROTATION CIPHER

The Rotation Cipher is implemented as ROT48 ('rotate by 48 places'). It is a simple character substitution cipher that replaces a character with the character 48 places after it in the ASCII table.

#### HOME.2.1. ROTATION CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 2

##### HOME.2.1.2. ENCRYPTION: ROTATION CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):  
This Is The Plain-Text!!

The Cipher Text Is:  
'%9:DPyDP%96P!=2:~]%6IEQQ'

#### HOME.2.1. ROTATION CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 3



## HOME.2.1.3. DECRYPTION: ROTATION CIPHER

Enter The Cipher Text (A String Of Length Greater Than 2):  
%9:DPyDP%96P!=2:~]6IEQQ

The Plain Text Is:  
'This Is The Plain-Text!!'

## HOME.2.1. ROTATION CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 4

## HOME.2. CIPHERS

1. Rotation Cipher
2. Atbash Cipher
3. Caesar Positive Shift Cipher
4. Caesar Negative Shift Cipher
5. One Time Pad Cipher
6. Monoalphabetic Substitution Cipher
7. Polyalphabetic Substitution Cipher
8. Autokey Cipher
9. Running Key Cipher
10. Polybius Square Cipher
11. Bifid Cipher
12. Trifid Cipher
13. Bitwise Rotation Cipher
14. Bitwise Atbash Cipher
15. Bitwise Binary Complement Cipher
16. Bitwise Left Shift Cipher
17. Bitwise Right Shift Cipher
18. Bitwise Exclusive Or Cipher
19. Bitwise One Time Pad Cipher
20. Square Transposition Cipher
21. Columnar Transposition Cipher
22. Double Transposition Cipher
23. Myszkowski Transposition Cipher
24. Chinese Transposition Cipher
25. Rail Fence Cipher
26. Back To HOME

Enter Choice: 2

## HOME.2.2. ATBASH CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 1

## HOME.2.2.1. ABOUT: ATBASH CIPHER

Atbash is a simple substitution cipher  
for the Hebrew alphabet. It consists

of substituting aleph (the first letter)  
for tav (the last), beth (the second)  
for shin (one before last), and so on,  
reversing the alphabet. The same  
algorithm is applied to the ASCII  
character.

#### HOME.2.2. ATBASH CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 2

#### HOME.2.2.2. ENCRYPTION: ATBASH CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):  
This Is The Plain-Text!!

The Cipher Text Is:  
'J65+~U+~J69~N2=50qJ9&\* } }'

#### HOME.2.2. ATBASH CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 3

#### HOME.2.2.3. DECRYPTION: ATBASH CIPHER

Enter The Cipher Text (A String Of Length Greater Than 2):  
J65+~U+~J69~N2=50qJ9&\* } }

The Plain Text Is:  
'This Is The Plain-Text!!'

#### HOME.2.2. ATBASH CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 4

#### HOME.2. CIPHERS

1. Rotation Cipher
2. Atbash Cipher
3. Caesar Positive Shift Cipher
4. Caesar Negative Shift Cipher
5. One Time Pad Cipher
6. Monoalphabetic Substitution Cipher
7. Polyalphabetic Substitution Cipher
8. Autokey Cipher
9. Running Key Cipher
10. Polybius Square Cipher

11. Bifid Cipher
  12. Trifid Cipher
  13. Bitwise Rotation Cipher
  14. Bitwise Atbash Cipher
  15. Bitwise Binary Compliment Cipher
  16. Bitwise Left Shift Cipher
  17. Bitwise Right Shift Cipher
  18. Bitwise Exclusive Or Cipher
  19. Bitwise One Time Pad Cipher
  20. Square Transposition Cipher
  21. Columnar Transposition Cipher
  22. Double Transposition Cipher
  23. Myszowski Transposition Cipher
  24. Chinese Transposition Cipher
  25. Rail Fence Cipher
  26. Back To HOME
- Enter Choice: 3

#### HOME.2.3. CAESAR POSITIVE SHIFT CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 1

##### HOME.2.3.1. ABOUT: CAESAR POSITIVE SHIFT CIPHER

The Caesar Positive Shift Cipher is implemented as a rotation cipher. It is a simple character substitution cipher that replaces a character with the character 'Key' number of places after it in the ASCII table. The value of 'Key' is taken as input from the user.

#### HOME.2.3. CAESAR POSITIVE SHIFT CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 2

##### HOME.2.3.2. ENCRYPTION: CAESAR POSITIVE SHIFT CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):  
This Is The Plain-Text!!

Enter the Key (A Number Between 0 and 10000):  
3

The Cipher Text Is:  
'Wklv#Lv#Wkh#Sodlq0Wh{w\$\$'

#### HOME.2.3. CAESAR POSITIVE SHIFT CIPHER

1. About
2. Encrypt

- 3. Decrypt
  - 4. Back To CIPHERS
- Enter Choice: 3

### HOME.2.3.3. DECRYPTION: CAESAR POSITIVE SHIFT CIPHER

Enter The Cipher Text (A String Of Length Greater Than 2):  
Wklv#Lv#Wkh#Sodlq0Wh{w\$\$

Enter the Key (A Number Between 0 and 10000):  
3

The Plain Text Is:  
'This Is The Plain-Text!!'

### HOME.2.3. CAESAR POSITIVE SHIFT CIPHER

- 1. About
  - 2. Encrypt
  - 3. Decrypt
  - 4. Back To CIPHERS
- Enter Choice: 4

### HOME.2. CIPHERS

- 1. Rotation Cipher
  - 2. Atbash Cipher
  - 3. Caesar Positive Shift Cipher
  - 4. Caesar Negative Shift Cipher
  - 5. One Time Pad Cipher
  - 6. Monoalphabetic Substitution Cipher
  - 7. Polyalphabetic Substitution Cipher
  - 8. Autokey Cipher
  - 9. Running Key Cipher
  - 10. Polybius Square Cipher
  - 11. Bifid Cipher
  - 12. Trifid Cipher
  - 13. Bitwise Rotation Cipher
  - 14. Bitwise Atbash Cipher
  - 15. Bitwise Binary Compliment Cipher
  - 16. Bitwise Left Shift Cipher
  - 17. Bitwise Right Shift Cipher
  - 18. Bitwise Exclusive Or Cipher
  - 19. Bitwise One Time Pad Cipher
  - 20. Square Transposition Cipher
  - 21. Columnar Transposition Cipher
  - 22. Double Transposition Cipher
  - 23. Myszkowski Transposition Cipher
  - 24. Chinese Transposition Cipher
  - 25. Rail Fence Cipher
  - 26. Back To HOME
- Enter Choice: 4

### HOME.2.4. CAESAR NEGATIVE SHIFT CIPHER

- 1. About
- 2. Encrypt
- 3. Decrypt

## 4. Back To CIPHERS

Enter Choice: 1

## HOME.2.4.1. ABOUT: CAESAR NEGATIVE SHIFT CIPHER

The Caesar Negative Shift Cipher is implemented as a rotation cipher. It is a simple character substitution cipher that replaces a character with the character 'Key' number of places before it in the ASCII table. The value of 'Key' is taken as input from the user.

## HOME.2.4. CAESAR NEGATIVE SHIFT CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 2

## HOME.2.4.2. ENCRYPTION: CAESAR NEGATIVE SHIFT CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):  
This Is The Plain-Text!!

Enter the Key (A Number Between 0 and 10000):  
3

The Cipher Text Is:  
'Qefp|Fp|Qeb|Mi^fk\*Qbuq} }'

## HOME.2.4. CAESAR NEGATIVE SHIFT CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 3

## HOME.2.4.3. DECRYPTION: CAESAR NEGATIVE SHIFT CIPHER

Enter The Cipher Text (A String Of Length Greater Than 2):  
Qefp|Fp|Qeb|Mi^fk\*Qbuq} }

Enter the Key (A Number Between 0 and 10000):  
3

The Plain Text Is:  
'This Is The Plain-Text!!'

## HOME.2.4. CAESAR NEGATIVE SHIFT CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 4

## HOME.2. CIPHERS

1. Rotation Cipher
2. Atbash Cipher
3. Caesar Positive Shift Cipher
4. Caesar Negative Shift Cipher
5. One Time Pad Cipher
6. Monoalphabetic Substitution Cipher
7. Polyalphabetic Substitution Cipher
8. Autokey Cipher
9. Running Key Cipher
10. Polybius Square Cipher
11. Bifid Cipher
12. Trifid Cipher
13. Bitwise Rotation Cipher
14. Bitwise Atbash Cipher
15. Bitwise Binary Compliment Cipher
16. Bitwise Left Shift Cipher
17. Bitwise Right Shift Cipher
18. Bitwise Exclusive Or Cipher
19. Bitwise One Time Pad Cipher
20. Square Transposition Cipher
21. Columnar Transposition Cipher
22. Double Transposition Cipher
23. Myszkowski Transposition Cipher
24. Chinese Transposition Cipher
25. Rail Fence Cipher
26. Back To HOME

Enter Choice: 5

## HOME.2.5. ONE TIME PAD CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 1

## HOME.2.5.1. ABOUT: ONE TIME PAD CIPHER

The One Time Pad Cipher is implemented as a rotation cipher. It is a simple character substitution cipher that replaces a character of the Plain Text by the number of places represented by the corresponding character in the 'Keyword'. For this the length of the 'Keyword' should be at least as long as the Plain Text. This is considered to be an unbreakable Cipher.

## HOME.2.5. ONE TIME PAD CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 2

#### HOME.2.5.2. ENCRYPTION: ONE TIME PAD CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):  
This Is The Plain-Text!!

Enter The Keyword (A String Of Length Atleast As Long As Plain Text):  
Computer Science Is My Favourite Subject.

The Cipher Text Is:  
'wXWdu>YrT<li6[EOnVHeFn!G'

#### HOME.2.5. ONE TIME PAD CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 3

#### HOME.2.5.3. DECRYPTION: ONE TIME PAD CIPHER

Enter The Cipher Text (A String Of Length Greater Than 2):  
wXWdu>YrT<li6[EOnVHeFn!G

Enter The Keyword (A String Of Length Atleast As Long As Plain Text):  
Computer Science Is My Favourite Subject.

The Plain Text Is:  
'This Is The Plain-Text!!'

#### HOME.2.5. ONE TIME PAD CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 4

#### HOME.2. CIPHERS

1. Rotation Cipher
2. Atbash Cipher
3. Caesar Positive Shift Cipher
4. Caesar Negative Shift Cipher
5. One Time Pad Cipher
6. Monoalphabetic Substitution Cipher
7. Polyalphabetic Substitution Cipher
8. Autokey Cipher
9. Running Key Cipher
10. Polybius Square Cipher
11. Bifid Cipher
12. Trifid Cipher
13. Bitwise Rotation Cipher
14. Bitwise Atbash Cipher
15. Bitwise Binary Compliment Cipher
16. Bitwise Left Shift Cipher

17. Bitwise Right Shift Cipher
18. Bitwise Exclusive Or Cipher
19. Bitwise One Time Pad Cipher
20. Square Transposition Cipher
21. Columnar Transposition Cipher
22. Double Transposition Cipher
23. Myszkowski Transposition Cipher
24. Chinese Transposition Cipher
25. Rail Fence Cipher
26. Back To HOME

Enter Choice: 6

#### HOME.2.6. MONOALPHABETIC SUBSTITUTION CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 1

##### HOME.2.6.1. ABOUT: MONOALPHABETIC SUBSTITUTION CIPHER

The Monoalphabetic Substitution Cipher is implemented as a substitution cipher where each character is replaced by another character based on the 'Key'. The 'Key' is prepared from the 'Keyword'.

#### HOME.2.6. MONOALPHABETIC SUBSTITUTION CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 2

##### HOME.2.6.2. ENCRYPTION: MONOALPHABETIC SUBSTITUTION CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):  
This Is The Plain-Text!!

Enter The Keyword (A String Of Length Greater Than 1):  
Computer

The Cipher Text Is:  
'MabnCAncMa^CIfZbh%M^xqoo'

#### HOME.2.6. MONOALPHABETIC SUBSTITUTION CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 3

##### HOME.2.6.3. DECRYPTION: MONOALPHABETIC SUBSTITUTION CIPHER

Enter The Cipher Text (A String Of Length Greater Than 2):



MabnCAncMa^ClfZbh%M^xqoo

Enter The Keyword (A String Of Length Greater Than 1):  
Computer

The Plain Text Is:  
'This Is The Plain-Text!!'

#### HOME.2.6. MONOALPHABETIC SUBSTITUTION CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 4

#### HOME.2. CIPHERS

1. Rotation Cipher
2. Atbash Cipher
3. Caesar Positive Shift Cipher
4. Caesar Negative Shift Cipher
5. One Time Pad Cipher
6. Monoalphabetic Substitution Cipher
7. Polyalphabetic Substitution Cipher
8. Autokey Cipher
9. Running Key Cipher
10. Polybius Square Cipher
11. Bifid Cipher
12. Trifid Cipher
13. Bitwise Rotation Cipher
14. Bitwise Atbash Cipher
15. Bitwise Binary Complement Cipher
16. Bitwise Left Shift Cipher
17. Bitwise Right Shift Cipher
18. Bitwise Exclusive Or Cipher
19. Bitwise One Time Pad Cipher
20. Square Transposition Cipher
21. Columnar Transposition Cipher
22. Double Transposition Cipher
23. Myszkowski Transposition Cipher
24. Chinese Transposition Cipher
25. Rail Fence Cipher
26. Back To HOME

Enter Choice: 7

#### HOME.2.7. POLYALPHABETIC SUBSTITUTION CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 1

##### HOME.2.7.1. ABOUT: POLYALPHABETIC SUBSTITUTION CIPHER

The Polyalphabetic Substitution Cipher is implemented  
as a substitution cipher where each character is

replaced by another character which is obtained by shifting the Plain Text character by the corresponding Keyword character.

#### HOME.2.7. POLYALPHABETIC SUBSTITUTION CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 2

##### HOME.2.7.2. ENCRYPTION: POLYALPHABETIC SUBSTITUTION CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):  
This Is The Plain-Text!!

Enter The Keyword (A String Of Length Greater Than 1):  
Computer

The Cipher Text Is:  
'wXWdu>YrwXSpFaG\2|BVnifs'

#### HOME.2.7. POLYALPHABETIC SUBSTITUTION CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 3

##### HOME.2.7.3. DECRYPTION: POLYALPHABETIC SUBSTITUTION CIPHER

Enter The Cipher Text (A String Of Length Greater Than 2):  
wXWdu>YrwXSpFaG\2|BVnifs

Enter The Keyword (A String Of Length Greater Than 1):  
Computer

The Plain Text Is:  
'This Is The Plain-Text!!'

#### HOME.2.7. POLYALPHABETIC SUBSTITUTION CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 4

#### HOME.2. CIPHERS

1. Rotation Cipher
2. Atbash Cipher
3. Caesar Positive Shift Cipher
4. Caesar Negative Shift Cipher
5. One Time Pad Cipher
6. Monoalphabetic Substitution Cipher
7. Polyalphabetic Substitution Cipher

8. Autokey Cipher
  9. Running Key Cipher
  10. Polybius Square Cipher
  11. Bifid Cipher
  12. Trifid Cipher
  13. Bitwise Rotation Cipher
  14. Bitwise Atbash Cipher
  15. Bitwise Binary Complement Cipher
  16. Bitwise Left Shift Cipher
  17. Bitwise Right Shift Cipher
  18. Bitwise Exclusive Or Cipher
  19. Bitwise One Time Pad Cipher
  20. Square Transposition Cipher
  21. Columnar Transposition Cipher
  22. Double Transposition Cipher
  23. Myszkowski Transposition Cipher
  24. Chinese Transposition Cipher
  25. Rail Fence Cipher
  26. Back To HOME
- Enter Choice: 8

#### HOME.2.8. AUTOKEY CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 1

##### HOME.2.8.1. ABOUT: AUTOKEY CIPHER

The Autokey Substitution Cipher is implemented as a substitution cipher where each character is replaced by another character which is obtained by shifting the Plain Text character by the corresponding 'Key' character. The 'Key' is obtained from the 'Keyword' by appending the Plain Text to the end of the Keyword.

#### HOME.2.8. AUTOKEY CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 2

##### HOME.2.8.2. ENCRYPTION: AUTOKEY CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):  
This Is The Plain-Text!!

Enter The Keyword (A String Of Length Greater Than 1):  
Computer

The Cipher Text Is:  
'wXWdu>Yr)QOsP6UiCu:eIabj'

#### HOME.2.8. AUTOKEY CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 3

#### HOME.2.8.3. DECRYPTION: AUTOKEY CIPHER

Enter The Cipher Text (A String Of Length Greater Than 2):  
wXWdu>Yr)QOsP6UiCu:eIabj

Enter The Keyword (A String Of Length Greater Than 1):  
Computer

The Plain Text Is:  
'This Is The Plain-Text!!'

#### HOME.2.8. AUTOKEY CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 4

#### HOME.2. CIPHERS

1. Rotation Cipher
2. Atbash Cipher
3. Caesar Positive Shift Cipher
4. Caesar Negative Shift Cipher
5. One Time Pad Cipher
6. Monoalphabetic Substitution Cipher
7. Polyalphabetic Substitution Cipher
8. Autokey Cipher
9. Running Key Cipher
10. Polybius Square Cipher
11. Bifid Cipher
12. Trifid Cipher
13. Bitwise Rotation Cipher
14. Bitwise Atbash Cipher
15. Bitwise Binary Compliment Cipher
16. Bitwise Left Shift Cipher
17. Bitwise Right Shift Cipher
18. Bitwise Exclusive Or Cipher
19. Bitwise One Time Pad Cipher
20. Square Transposition Cipher
21. Columnar Transposition Cipher
22. Double Transposition Cipher
23. Myszkowski Transposition Cipher
24. Chinese Transposition Cipher
25. Rail Fence Cipher
26. Back To HOME

Enter Choice: 9

#### HOME.2.9. RUNNING KEY CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 1

##### HOME.2.9.1. ABOUT: RUNNING KEY CIPHER

The Running Key Cipher is implemented as a substitution cipher where each character is replaced by another character which is obtained by shifting the Plain Text character by the corresponding Keyword character. For this the length of the 'Keyword' should be at least as long as the Plain Text.

#### HOME.2.9. RUNNING KEY CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 2

##### HOME.2.9.2. ENCRYPTION: RUNNING KEY CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):  
This Is The Plain-Text!!

Enter The Keyword (A String Of Length Atleast As Long As Plain Text):  
Computer Science Is My Favourite Subject.

The Cipher Text Is:  
'wXWdu>YrT<li6[EO nVHeFn!G'

#### HOME.2.9. RUNNING KEY CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 3

##### HOME.2.9.3. DECRYPTION: RUNNING KEY CIPHER

Enter The Cipher Text (A String Of Length Greater Than 2):  
wXWdu>YrT<li6[EO nVHeFn!G

Enter The Keyword (A String Of Length Atleast As Long As Plain Text):  
Computer Science Is My Favourite Subject.

The Plain Text Is:  
'This Is The Plain-Text!!'

## HOME.2.9. RUNNING KEY CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 4

## HOME.2. CIPHERS

1. Rotation Cipher
2. Atbash Cipher
3. Caesar Positive Shift Cipher
4. Caesar Negative Shift Cipher
5. One Time Pad Cipher
6. Monoalphabetic Substitution Cipher
7. Polyalphabetic Substitution Cipher
8. Autokey Cipher
9. Running Key Cipher
10. Polybius Square Cipher
11. Bifid Cipher
12. Trifid Cipher
13. Bitwise Rotation Cipher
14. Bitwise Atbash Cipher
15. Bitwise Binary Compliment Cipher
16. Bitwise Left Shift Cipher
17. Bitwise Right Shift Cipher
18. Bitwise Exclusive Or Cipher
19. Bitwise One Time Pad Cipher
20. Square Transposition Cipher
21. Columnar Transposition Cipher
22. Double Transposition Cipher
23. Myszkowski Transposition Cipher
24. Chinese Transposition Cipher
25. Rail Fence Cipher
26. Back To HOME

Enter Choice: 10

## HOME.2.10. POLYBIUS SQUARE CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 1

## HOME.2.10.1. ABOUT: POLYBIUS SQUARE CIPHER

The Polybius Square Cipher makes use of the Polybius square. The square has 10 rows and 10 columns each numbered from 0 through 9. Each element in the square is a character. It is used to substitute the characters for numbers corresponding to their location in the square in the format 'row number', 'column number'. The Cipher Text generated by the program is a sequence of 3 digit ASCII values

of the Cipher Text characters.

#### HOME.2.10. POLYBIUS SQUARE CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 2

#### HOME.2.10.2. ENCRYPTION: POLYBIUS SQUARE CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):  
This Is The Plain-Text!!

The Cipher Text Is:  
'527273830041830052726900487665737813526988840101'

#### HOME.2.10. POLYBIUS SQUARE CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 3

#### HOME.2.10.3. DECRYPTION: POLYBIUS SQUARE CIPHER

Enter The Cipher Text (A Sequence Of Numbers Of Length Greater Than 4 And Of  
Length A Multiple Of 2):

527273830041830052726900487665737813526988840101

The Plain Text Is:  
'This Is The Plain-Text!!'

#### HOME.2.10. POLYBIUS SQUARE CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 4

#### HOME.2. CIPHERS

1. Rotation Cipher
2. Atbash Cipher
3. Caesar Positive Shift Cipher
4. Caesar Negative Shift Cipher
5. One Time Pad Cipher
6. Monoalphabetic Substitution Cipher
7. Polyalphabetic Substitution Cipher
8. Autokey Cipher
9. Running Key Cipher
10. Polybius Square Cipher
11. Bifid Cipher
12. Trifid Cipher
13. Bitwise Rotation Cipher
14. Bitwise Atbash Cipher

15. Bitwise Binary Compliment Cipher
  16. Bitwise Left Shift Cipher
  17. Bitwise Right Shift Cipher
  18. Bitwise Exclusive Or Cipher
  19. Bitwise One Time Pad Cipher
  20. Square Transposition Cipher
  21. Columnar Transposition Cipher
  22. Double Transposition Cipher
  23. Myszkowski Transposition Cipher
  24. Chinese Transposition Cipher
  25. Rail Fence Cipher
  26. Back To HOME
- Enter Choice: 11

#### HOME.2.11. BIFID CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 1

##### HOME.2.11.1. ABOUT: BIFID CIPHER

The Bifid Cipher makes use of the Polybius square. The square has 10 rows and 10 columns each numbered from 0 through 9. Each element in the square is a character. It is used to substitute the characters for numbers corresponding to their location in the square while writing all the row numbers first followed by all the column numbers. The Cipher Text generated by the program is a sequence of 3 digit ASCII values of the Cipher Text characters.

#### HOME.2.11. BIFID CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 2

##### HOME.2.11.2. ENCRYPTION: BIFID CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):  
This Is The Plain-Text!!

The Cipher Text Is:  
'577804805760476771568800223301302290865383298411'

#### HOME.2.11. BIFID CIPHER

1. About
2. Encrypt



- 3. Decrypt
  - 4. Back To CIPHERS
- Enter Choice: 3

### HOME.2.11.3. DECRYPTION: BIFID CIPHER

Enter The Cipher Text (A Sequence Of Numbers Of Length Greater Than 4 And Of Length A Multiple Of 2):

577804805760476771568800223301302290865383298411

The Plain Text Is:  
'This Is The Plain-Text!!'

### HOME.2.11. BIFID CIPHER

- 1. About
  - 2. Encrypt
  - 3. Decrypt
  - 4. Back To CIPHERS
- Enter Choice: 4

### HOME.2. CIPHERS

- 1. Rotation Cipher
  - 2. Atbash Cipher
  - 3. Caesar Positive Shift Cipher
  - 4. Caesar Negative Shift Cipher
  - 5. One Time Pad Cipher
  - 6. Monoalphabetic Substitution Cipher
  - 7. Polyalphabetic Substitution Cipher
  - 8. Autokey Cipher
  - 9. Running Key Cipher
  - 10. Polybius Square Cipher
  - 11. Bifid Cipher
  - 12. Trifid Cipher
  - 13. Bitwise Rotation Cipher
  - 14. Bitwise Atbash Cipher
  - 15. Bitwise Binary Complement Cipher
  - 16. Bitwise Left Shift Cipher
  - 17. Bitwise Right Shift Cipher
  - 18. Bitwise Exclusive Or Cipher
  - 19. Bitwise One Time Pad Cipher
  - 20. Square Transposition Cipher
  - 21. Columnar Transposition Cipher
  - 22. Double Transposition Cipher
  - 23. Myszkowski Transposition Cipher
  - 24. Chinese Transposition Cipher
  - 25. Rail Fence Cipher
  - 26. Back To HOME
- Enter Choice: 12

### HOME.2.12. TRIFID CIPHER

- 1. About
  - 2. Encrypt
  - 3. Decrypt
  - 4. Back To CIPHERS
- Enter Choice: 1

## HOME.2.12.1. ABOUT: TRIFID CIPHER

The Trifid Cipher uses a cuboid of size 3 X 6 X 6. It can be viewed as 3 squares (numbered from 0 through 2) with 6 rows (numbered from 0 through 5) and 6 columns (numbered from 0 through 5). Each element in the cuboid is a character. It is used to substitute the characters for numbers corresponding to their location in the cuboid by writing all the square numbers first, all the row numbers second followed by all the column numbers third. The Cipher Text generated by the program is a sequence of 3 digit ASCII values of the Cipher Text characters.

## HOME.2.12. TRIFID CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 2

## HOME.2.12.2. ENCRYPTION: TRIFID CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):  
This Is The Plain-Text!!

The Cipher Text Is:

'122201201210121220112200200100102050204012252200401505504030045101434011'

## HOME.2.12. TRIFID CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 3

## HOME.2.12.3. DECRYPTION: TRIFID CIPHER

Enter The Cipher Text (A Sequence Of Numbers Of Length Greater Than 6 And Of Length A Multiple Of 3):

122201201210121220112200200100102050204012252200401505504030045101434011

The Plain Text Is:  
'This Is The Plain-Text!!'

## HOME.2.12. TRIFID CIPHER

1. About
2. Encrypt

3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 4

## HOME.2. CIPHERS

1. Rotation Cipher
  2. Atbash Cipher
  3. Caesar Positive Shift Cipher
  4. Caesar Negative Shift Cipher
  5. One Time Pad Cipher
  6. Monoalphabetic Substitution Cipher
  7. Polyalphabetic Substitution Cipher
  8. Autokey Cipher
  9. Running Key Cipher
  10. Polybius Square Cipher
  11. Bifid Cipher
  12. Trifid Cipher
  13. Bitwise Rotation Cipher
  14. Bitwise Atbash Cipher
  15. Bitwise Binary Complement Cipher
  16. Bitwise Left Shift Cipher
  17. Bitwise Right Shift Cipher
  18. Bitwise Exclusive Or Cipher
  19. Bitwise One Time Pad Cipher
  20. Square Transposition Cipher
  21. Columnar Transposition Cipher
  22. Double Transposition Cipher
  23. Myszowski Transposition Cipher
  24. Chinese Transposition Cipher
  25. Rail Fence Cipher
  26. Back To HOME
- Enter Choice: 13

### HOME.2.13. BITWISE ROTATION CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 1

#### HOME.2.13.1. ABOUT: BITWISE ROTATION CIPHER

In the Bitwise Rotation Cipher the rotation cipher is applied at the bit level. All the ASCII characters can be represented by 8 bits (1 byte). This gives us  $2^8 = 256$  possible characters. The Cipher Text generated by the program is a sequence of 3 digit ASCII values of the Cipher Text characters.

### HOME.2.13. BITWISE ROTATION CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 2

HOME.2.13.2. ENCRYPTION: BITWISE ROTATION CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):  
This Is The Plain-Text!!

The Cipher Text Is:

'069134150055002148055002069134086002005198022150230210069086135071018018'

HOME.2.13. BITWISE ROTATION CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 3

HOME.2.13.3. DECRYPTION: BITWISE ROTATION CIPHER

Enter The Cipher Text (A Sequence Of Numbers Of Length Greater Than 6 And Of Length A Multiple Of 3):

069134150055002148055002069134086002005198022150230210069086135071018018

The Plain Text Is:  
'This Is The Plain-Text!!'

HOME.2.13. BITWISE ROTATION CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 4

HOME.2. CIPHERS

1. Rotation Cipher
2. Atbash Cipher
3. Caesar Positive Shift Cipher
4. Caesar Negative Shift Cipher
5. One Time Pad Cipher
6. Monoalphabetic Substitution Cipher
7. Polyalphabetic Substitution Cipher
8. Autokey Cipher
9. Running Key Cipher
10. Polybius Square Cipher
11. Bifid Cipher
12. Trifid Cipher
13. Bitwise Rotation Cipher
14. Bitwise Atbash Cipher
15. Bitwise Binary Complement Cipher
16. Bitwise Left Shift Cipher
17. Bitwise Right Shift Cipher
18. Bitwise Exclusive Or Cipher
19. Bitwise One Time Pad Cipher

- 20. Square Transposition Cipher
- 21. Columnar Transposition Cipher
- 22. Double Transposition Cipher
- 23. Myszkowski Transposition Cipher
- 24. Chinese Transposition Cipher
- 25. Rail Fence Cipher
- 26. Back To HOME

Enter Choice: 14

#### HOME.2.14. BITWISE ATBASH CIPHER

- 1. About
- 2. Encrypt
- 3. Decrypt
- 4. Back To CIPHERS

Enter Choice: 1

##### HOME.2.14.1. ABOUT: BITWISE ATBASH CIPHER

In the Bitwise Atbash Cipher the atbash cipher is applied at the bit level. All the ASCII characters can be represented by 8 bits (1 byte). This gives us  $2^8 = 256$  possible characters. The Cipher Text generated by the program is a sequence of 3 digit ASCII values of the Cipher Text characters.

#### HOME.2.14. BITWISE ATBASH CIPHER

- 1. About
- 2. Encrypt
- 3. Decrypt
- 4. Back To CIPHERS

Enter Choice: 2

##### HOME.2.14.2. ENCRYPTION: BITWISE ATBASH CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):  
This Is The Plain-Text!!

The Cipher Text Is:

'042022150206004146206004042022166004010054134150118180042166030046132132'

#### HOME.2.14. BITWISE ATBASH CIPHER

- 1. About
- 2. Encrypt
- 3. Decrypt
- 4. Back To CIPHERS

Enter Choice: 3

##### HOME.2.14.3. DECRYPTION: BITWISE ATBASH CIPHER

Enter The Cipher Text (A Sequence Of Numbers Of Length Greater Than 6 And Of Length A Multiple Of 3):

042022150206004146206004042022166004010054134150118180042166030046132132

The Plain Text Is:  
'This Is The Plain-Text!!'

#### HOME.2.14. BITWISE ATBASH CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 4

#### HOME.2. CIPHERS

1. Rotation Cipher
2. Atbash Cipher
3. Caesar Positive Shift Cipher
4. Caesar Negative Shift Cipher
5. One Time Pad Cipher
6. Monoalphabetic Substitution Cipher
7. Polyalphabetic Substitution Cipher
8. Autokey Cipher
9. Running Key Cipher
10. Polybius Square Cipher
11. Bifid Cipher
12. Trifid Cipher
13. Bitwise Rotation Cipher
14. Bitwise Atbash Cipher
15. Bitwise Binary Compliment Cipher
16. Bitwise Left Shift Cipher
17. Bitwise Right Shift Cipher
18. Bitwise Exclusive Or Cipher
19. Bitwise One Time Pad Cipher
20. Square Transposition Cipher
21. Columnar Transposition Cipher
22. Double Transposition Cipher
23. Myszkowski Transposition Cipher
24. Chinese Transposition Cipher
25. Rail Fence Cipher
26. Back To HOME

Enter Choice: 15

#### HOME.2.15. BITWISE BINARY COMPLIMENT CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 1

##### HOME.2.15.1. ABOUT: BITWISE BINARY COMPLIMENT CIPHER

In the Bitwise Binary Compliment Cipher each bit is complimented, i.e., 1 becomes 0 and 0 becomes 1. All the ASCII characters can be represented by 8 bits (1 byte). This gives us  $2^8 = 256$  possible characters. The

Cipher Text generated by the program is a sequence of 3 digit ASCII values of the Cipher Text characters.

#### HOME.2.15. BITWISE BINARY COMPLIMENT CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 2

#### HOME.2.15.2. ENCRYPTION: BITWISE BINARY COMPLIMENT CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):  
This Is The Plain-Text!!

The Cipher Text Is:

'171151150140223182140223171151154223175147158150145210171154135139222222'

#### HOME.2.15. BITWISE BINARY COMPLIMENT CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 3

#### HOME.2.15.3. DECRYPTION: BITWISE BINARY COMPLIMENT CIPHER

Enter The Cipher Text (A Sequence Of Numbers Of Length Greater Than 6 And Of Length A Multiple Of 3):

171151150140223182140223171151154223175147158150145210171154135139222222

The Plain Text Is:  
'This Is The Plain-Text!!'

#### HOME.2.15. BITWISE BINARY COMPLIMENT CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 4

#### HOME.2. CIPHERS

1. Rotation Cipher
2. Atbash Cipher
3. Caesar Positive Shift Cipher
4. Caesar Negative Shift Cipher
5. One Time Pad Cipher
6. Monoalphabetic Substitution Cipher
7. Polyalphabetic Substitution Cipher
8. Autokey Cipher
9. Running Key Cipher
10. Polybius Square Cipher
11. Bifid Cipher

12. Trifid Cipher
  13. Bitwise Rotation Cipher
  14. Bitwise Atbash Cipher
  15. Bitwise Binary Compliment Cipher
  16. Bitwise Left Shift Cipher
  17. Bitwise Right Shift Cipher
  18. Bitwise Exclusive Or Cipher
  19. Bitwise One Time Pad Cipher
  20. Square Transposition Cipher
  21. Columnar Transposition Cipher
  22. Double Transposition Cipher
  23. Myszkowski Transposition Cipher
  24. Chinese Transposition Cipher
  25. Rail Fence Cipher
  26. Back To HOME
- Enter Choice: 16

#### HOME.2.16. BITWISE LEFT SHIFT CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 1

##### HOME.2.16.1. ABOUT: BITWISE LEFT SHIFT CIPHER

In the Bitwise Left Shift Cipher each bit is shifted to the left by the number in the Keyword (0-7). All the ASCII characters can be represented by 8 bits (1 byte). This gives us  $2^8 = 256$  possible characters. The Cipher Text generated by the program is a sequence of 3 digit ASCII values of the Cipher Text characters.

#### HOME.2.16. BITWISE LEFT SHIFT CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 2

##### HOME.2.16.2. ENCRYPTION: BITWISE LEFT SHIFT CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):  
This Is The Plain-Text!!

Enter the Key (A Number In The Range Of 0 Through 7):  
3

The Cipher Text Is:

'162067075155001074155001162067043001130099011075115105162043195163009009'

#### HOME.2.16. BITWISE LEFT SHIFT CIPHER

1. About



2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 3

### HOME.2.16.3. DECRYPTION: BITWISE LEFT SHIFT CIPHER

Enter The Cipher Text (A Sequence Of Numbers Of Length Greater Than 6 And Of Length A Multiple Of 3):

162067075155001074155001162067043001130099011075115105162043195163009009

Enter the Key (A Number In The Range Of 0 Through 7):  
3

The Plain Text Is:  
'This Is The Plain-Text!!'

### HOME.2.16. BITWISE LEFT SHIFT CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 4

### HOME.2. CIPHERS

1. Rotation Cipher
  2. Atbash Cipher
  3. Caesar Positive Shift Cipher
  4. Caesar Negative Shift Cipher
  5. One Time Pad Cipher
  6. Monoalphabetic Substitution Cipher
  7. Polyalphabetic Substitution Cipher
  8. Autokey Cipher
  9. Running Key Cipher
  10. Polybius Square Cipher
  11. Bifid Cipher
  12. Trifid Cipher
  13. Bitwise Rotation Cipher
  14. Bitwise Atbash Cipher
  15. Bitwise Binary Compliment Cipher
  16. Bitwise Left Shift Cipher
  17. Bitwise Right Shift Cipher
  18. Bitwise Exclusive Or Cipher
  19. Bitwise One Time Pad Cipher
  20. Square Transposition Cipher
  21. Columnar Transposition Cipher
  22. Double Transposition Cipher
  23. Myszkowski Transposition Cipher
  24. Chinese Transposition Cipher
  25. Rail Fence Cipher
  26. Back To HOME
- Enter Choice: 17

### HOME.2.17. BITWISE RIGHT SHIFT CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 1

#### HOME.2.17.1. ABOUT: BITWISE RIGHT SHIFT CIPHER

In the Bitwise Right Shift Cipher each bit is shifted to the right by the number in the Keyword (0-7). All the ASCII characters can be represented by 8 bits (1 byte). This gives us  $2^8 = 256$  possible characters. The Cipher Text generated by the program is a sequence of 3 digit ASCII values of the Cipher Text characters.

#### HOME.2.17. BITWISE RIGHT SHIFT CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 2

#### HOME.2.17.2. ENCRYPTION: BITWISE RIGHT SHIFT CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):  
This Is The Plain-Text!!

Enter the Key (A Number In The Range Of 0 Through 7):  
3

The Cipher Text Is:

'138013045110004041110004138013172004010141044045205165138172015142036036'

#### HOME.2.17. BITWISE RIGHT SHIFT CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 3

#### HOME.2.17.3. DECRYPTION: BITWISE RIGHT SHIFT CIPHER

Enter The Cipher Text (A Sequence Of Numbers Of Length Greater Than 6 And Of Length A Multiple Of 3):

138013045110004041110004138013172004010141044045205165138172015142036036

Enter the Key (A Number In The Range Of 0 Through 7):  
3

The Plain Text Is:  
'This Is The Plain-Text!!'

## HOME.2.17. BITWISE RIGHT SHIFT CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 4

## HOME.2. CIPHERS

1. Rotation Cipher
  2. Atbash Cipher
  3. Caesar Positive Shift Cipher
  4. Caesar Negative Shift Cipher
  5. One Time Pad Cipher
  6. Monoalphabetic Substitution Cipher
  7. Polyalphabetic Substitution Cipher
  8. Autokey Cipher
  9. Running Key Cipher
  10. Polybius Square Cipher
  11. Bifid Cipher
  12. Trifid Cipher
  13. Bitwise Rotation Cipher
  14. Bitwise Atbash Cipher
  15. Bitwise Binary Compliment Cipher
  16. Bitwise Left Shift Cipher
  17. Bitwise Right Shift Cipher
  18. Bitwise Exclusive Or Cipher
  19. Bitwise One Time Pad Cipher
  20. Square Transposition Cipher
  21. Columnar Transposition Cipher
  22. Double Transposition Cipher
  23. Myszkowski Transposition Cipher
  24. Chinese Transposition Cipher
  25. Rail Fence Cipher
  26. Back To HOME
- Enter Choice: 18

## HOME.2.18. BITWISE EXCLUSIVE OR CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 1

## HOME.2.18.1. ABOUT: BITWISE EXCLUSIVE OR CIPHER

In the Bitwise Exclusive Or Cipher the logical XOR (Exclusive Or) operator is used. The corresponding bits of the Keyword and the Plain Text on applying the XOR operation give the Cipher Text. All the ASCII characters can be represented by 8 bits (1 byte). This gives us  $2^8 = 256$  possible characters. The Cipher Text generated by the program is a sequence of 3 digit ASCII values of the Cipher Text characters.

## HOME.2.18. BITWISE EXCLUSIVE OR CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 2

## HOME.2.18.2. ENCRYPTION: BITWISE EXCLUSIVE OR CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):  
This Is The Plain-Text!!

Enter The Keyword (A String Of Length Greater Than 1):  
Computer

The Cipher Text Is:

'023007004003085061022082023007008080037024004027045066057021013000068083'

## HOME.2.18. BITWISE EXCLUSIVE OR CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 3

## HOME.2.18.3. DECRYPTION: BITWISE EXCLUSIVE OR CIPHER

Enter The Cipher Text (A Sequence Of Numbers Of Length Greater Than 6 And Of Length A Multiple Of 3):

023007004003085061022082023007008080037024004027045066057021013000068083

Enter The Keyword (A String Of Length Greater Than 1):  
Computer

The Plain Text Is:  
'This Is The Plain-Text!!'

## HOME.2.18. BITWISE EXCLUSIVE OR CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 4

## HOME.2. CIPHERS

1. Rotation Cipher
2. Atbash Cipher
3. Caesar Positive Shift Cipher
4. Caesar Negative Shift Cipher
5. One Time Pad Cipher
6. Monoalphabetic Substitution Cipher
7. Polyalphabetic Substitution Cipher
8. Autokey Cipher

9. Running Key Cipher
10. Polybius Square Cipher
11. Bifid Cipher
12. Trifid Cipher
13. Bitwise Rotation Cipher
14. Bitwise Atbash Cipher
15. Bitwise Binary Complement Cipher
16. Bitwise Left Shift Cipher
17. Bitwise Right Shift Cipher
18. Bitwise Exclusive Or Cipher
19. Bitwise One Time Pad Cipher
20. Square Transposition Cipher
21. Columnar Transposition Cipher
22. Double Transposition Cipher
23. Myszkowski Transposition Cipher
24. Chinese Transposition Cipher
25. Rail Fence Cipher
26. Back To HOME

Enter Choice: 19

#### HOME.2.19. BITWISE ONE TIME PAD CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 1

##### HOME.2.19.1. ABOUT: BITWISE ONE TIME PAD CIPHER

In the Bitwise One Time Pad Cipher the logical XOR (Exclusive Or) operator is used. The corresponding bits of the Keyword and the Plain Text on applying the XOR operation give the Cipher Text. The Keyword must be at least as long as the Plain Text. All the ASCII characters can be represented by 8 bits (1 byte). This gives us  $2^8 = 256$  possible characters. The Cipher Text generated by the program is a sequence of 3 digit ASCII values of the Cipher Text characters.

#### HOME.2.19. BITWISE ONE TIME PAD CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 2

##### HOME.2.19.2. ENCRYPTION: BITWISE ONE TIME PAD CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):  
This Is The Plain-Text!!

Enter The Keyword (A String Of Length Atleast As Long As Plain Text):  
Computer Science Is My Favourite Subject.

The Cipher Text Is:

'023007004003085061022082116059006073053002002012078100039069053013001103'

HOME.2.19. BITWISE ONE TIME PAD CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 3

HOME.2.19.3. DECRYPTION: BITWISE ONE TIME PAD CIPHER

Enter The Cipher Text (A Sequence Of Numbers Of Length Greater Than 6 And Of Length A Multiple Of 3):

023007004003085061022082116059006073053002002012078100039069053013001103

Enter The Keyword (A String Of Length Atleast As Long As One Third Of Cipher Text):

Computer Science Is My Favourite Subject.

The Plain Text Is:

'This Is The Plain-Text!!'

HOME.2.19. BITWISE ONE TIME PAD CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 4

HOME.2. CIPHERS

1. Rotation Cipher
2. Atbash Cipher
3. Caesar Positive Shift Cipher
4. Caesar Negative Shift Cipher
5. One Time Pad Cipher
6. Monoalphabetic Substitution Cipher
7. Polyalphabetic Substitution Cipher
8. Autokey Cipher
9. Running Key Cipher
10. Polybius Square Cipher
11. Bifid Cipher
12. Trifid Cipher
13. Bitwise Rotation Cipher
14. Bitwise Atbash Cipher
15. Bitwise Binary Compliment Cipher
16. Bitwise Left Shift Cipher
17. Bitwise Right Shift Cipher
18. Bitwise Exclusive Or Cipher
19. Bitwise One Time Pad Cipher
20. Square Transposition Cipher
21. Columnar Transposition Cipher
22. Double Transposition Cipher

- 23. Myszowski Transposition Cipher
- 24. Chinese Transposition Cipher
- 25. Rail Fence Cipher
- 26. Back To HOME

Enter Choice: 20

#### HOME.2.20. SQUARE TRANSPOSITION CIPHER

- 1. About
- 2. Encrypt
- 3. Decrypt
- 4. Back To CIPHERS

Enter Choice: 1

##### HOME.2.20.1. ABOUT: SQUARE TRANSPOSITION CIPHER

The Square Transposition cipher arranges the Plain Text in a perfect square, i.e., number of rows equal the number of columns, where each element in the square has a character. If there are empty elements they are padded with spaces. The cipher is obtained by interchanging the rows with the columns.

#### HOME.2.20. SQUARE TRANSPOSITION CIPHER

- 1. About
- 2. Encrypt
- 3. Decrypt
- 4. Back To CIPHERS

Enter Choice: 2

##### HOME.2.20.2. ENCRYPTION: SQUARE TRANSPOSITION CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):  
This Is The Plain-Text!!

The Cipher Text Is:  
'Tleixhs nti P-!sTIT! hae '

#### HOME.2.20. SQUARE TRANSPOSITION CIPHER

- 1. About
- 2. Encrypt
- 3. Decrypt
- 4. Back To CIPHERS

Enter Choice: 3

##### HOME.2.20.3. DECRYPTION: SQUARE TRANSPOSITION CIPHER

Enter The Cipher Text (A String Of Length Greater Than 2):  
Tleixhs nti P-!sTIT! hae

The Plain Text Is:  
'This Is The Plain-Text!!'

#### HOME.2.20. SQUARE TRANSPOSITION CIPHER

- 1. About

2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 4

## HOME.2. CIPHERS

1. Rotation Cipher
  2. Atbash Cipher
  3. Caesar Positive Shift Cipher
  4. Caesar Negative Shift Cipher
  5. One Time Pad Cipher
  6. Monoalphabetic Substitution Cipher
  7. Polyalphabetic Substitution Cipher
  8. Autokey Cipher
  9. Running Key Cipher
  10. Polybius Square Cipher
  11. Bifid Cipher
  12. Trifid Cipher
  13. Bitwise Rotation Cipher
  14. Bitwise Atbash Cipher
  15. Bitwise Binary Compliment Cipher
  16. Bitwise Left Shift Cipher
  17. Bitwise Right Shift Cipher
  18. Bitwise Exclusive Or Cipher
  19. Bitwise One Time Pad Cipher
  20. Square Transposition Cipher
  21. Columnar Transposition Cipher
  22. Double Transposition Cipher
  23. Myszkowski Transposition Cipher
  24. Chinese Transposition Cipher
  25. Rail Fence Cipher
  26. Back To HOME
- Enter Choice: 21

### HOME.2.21. COLUMNAR TRANSPOSITION CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 1

#### HOME.2.21.1. ABOUT: COLUMNAR TRANSPOSITION CIPHER

In a columnar transposition, the message is written out in rows of a fixed length, and then read out again column by column, and the columns are chosen in some scrambled order. Both the width of the rows and the permutation of the columns are based on the Keyword.

### HOME.2.21. COLUMNAR TRANSPOSITION CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS



Enter Choice: 2

#### HOME.2.21.2. ENCRYPTION: COLUMNAR TRANSPOSITION CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):  
This Is The Plain-Text!!

Enter The Keyword (A String Of Length Greater Than 1 And Less Than 100 And  
Less Than Length Of Plain Text):  
Computer

The Cipher Text Is:  
'TTnsa!ieThh-s e i!lIt Px'

#### HOME.2.21. COLUMNAR TRANSPOSITION CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 3

#### HOME.2.21.3. DECRYPTION: COLUMNAR TRANSPOSITION CIPHER

Enter The Cipher Text (A String Of Length Greater Than 2):  
TTnsa!ieThh-s e i!lIt Px

Enter The Keyword (A String Of Length Greater Than 1 And Less Than 100 And  
Less Than Length Of Cipher Text):  
Computer

The Plain Text Is:  
'This Is The Plain-Text!!'

#### HOME.2.21. COLUMNAR TRANSPOSITION CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 4

#### HOME.2. CIPHERS

1. Rotation Cipher
2. Atbash Cipher
3. Caesar Positive Shift Cipher
4. Caesar Negative Shift Cipher
5. One Time Pad Cipher
6. Monoalphabetic Substitution Cipher
7. Polyalphabetic Substitution Cipher
8. Autokey Cipher
9. Running Key Cipher
10. Polybius Square Cipher
11. Bifid Cipher
12. Trifid Cipher
13. Bitwise Rotation Cipher
14. Bitwise Atbash Cipher

15. Bitwise Binary Compliment Cipher
16. Bitwise Left Shift Cipher
17. Bitwise Right Shift Cipher
18. Bitwise Exclusive Or Cipher
19. Bitwise One Time Pad Cipher
20. Square Transposition Cipher
21. Columnar Transposition Cipher
22. Double Transposition Cipher
23. Myszkowski Transposition Cipher
24. Chinese Transposition Cipher
25. Rail Fence Cipher
26. Back To HOME

Enter Choice: 22

#### HOME.2.22. DOUBLE TRANSPOSITION CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 1

##### HOME.2.22.1. ABOUT: DOUBLE TRANSPOSITION CIPHER

The Double Transposition Cipher is simply the Columnar Transposition Cipher applied twice. The same Keyword can be used for both transpositions or two different Keywords can be used.

#### HOME.2.22. DOUBLE TRANSPOSITION CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 2

##### HOME.2.22.2. ENCRYPTION: DOUBLE TRANSPOSITION CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):  
This Is The Plain-Text!!

Enter The First Keyword (A String Of Length Greater Than 1 And Less Than 100  
And Less Than Length Of Plain Text):  
Computer

Enter The Second Keyword (A String Of Length Greater Than 1 And Less Than  
100 And Less Than Length Of Plain Text):  
Project

The Cipher Text Is:  
'Tee !sl a-I sh! nhixTT Pi t '

#### HOME.2.22. DOUBLE TRANSPOSITION CIPHER

1. About
2. Encrypt

- 3. Decrypt
  - 4. Back To CIPHERS
- Enter Choice: 3

### HOME.2.22.3. DECRYPTION: DOUBLE TRANSPOSITION CIPHER

Enter The Cipher Text (A String Of Length Greater Than 2):  
Tee !sl a-I sh! nhixTT Pi t

Enter The Second Keyword (A String Of Length Greater Than 1 And Less Than 100 And Less Than Length Of Cipher Text):  
Project

Enter The First Keyword (A String Of Length Greater Than 1 And Less Than 100 And Less Than Length Of Cipher Text):  
Computer

The Plain Text Is:  
'This Is The Plain-Text!!'

### HOME.2.22. DOUBLE TRANSPOSITION CIPHER

- 1. About
  - 2. Encrypt
  - 3. Decrypt
  - 4. Back To CIPHERS
- Enter Choice: 4

### HOME.2. CIPHERS

- 1. Rotation Cipher
- 2. Atbash Cipher
- 3. Caesar Positive Shift Cipher
- 4. Caesar Negative Shift Cipher
- 5. One Time Pad Cipher
- 6. Monoalphabetic Substitution Cipher
- 7. Polyalphabetic Substitution Cipher
- 8. Autokey Cipher
- 9. Running Key Cipher
- 10. Polybius Square Cipher
- 11. Bifid Cipher
- 12. Trifid Cipher
- 13. Bitwise Rotation Cipher
- 14. Bitwise Atbash Cipher
- 15. Bitwise Binary Compliment Cipher
- 16. Bitwise Left Shift Cipher
- 17. Bitwise Right Shift Cipher
- 18. Bitwise Exclusive Or Cipher
- 19. Bitwise One Time Pad Cipher
- 20. Square Transposition Cipher
- 21. Columnar Transposition Cipher
- 22. Double Transposition Cipher
- 23. Myszkowski Transposition Cipher
- 24. Chinese Transposition Cipher
- 25. Rail Fence Cipher
- 26. Back To HOME

Enter Choice: 23

#### HOME.2.23. MYSZKOWSKI TRANSPOSITION CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 1

##### HOME.2.23.1. ABOUT: MYSZKOWSKI TRANSPOSITION CIPHER

The Myszowski Transposition is a variant form of Columnar Transposition, proposed by Émile Victor Théodore Myszowski in 1902. It requires a keyword with recurrent letters. In Columnar Transposition subsequent occurrences of a keyword letter are treated as if the next letter in alphabetical order, e.g., the keyword TOMATO yields a numeric keysting of '532164'. In Myszowski Transposition, recurrent keyword letters are numbered identically, thus TOMATO yielding a keysting of '432143.'

#### HOME.2.23. MYSZKOWSKI TRANSPOSITION CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 2

##### HOME.2.23.2. ENCRYPTION: MYSZKOWSKI TRANSPOSITION CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):  
This Is The Plain-Text!!

Enter The Keyword (A String Of Length Greater Than 1 And Less Than 100 And Less Than Length Of Plain Text):  
Computer

The Cipher Text Is:  
'TTnsa!ieThh-s e i!lIt Px'

#### HOME.2.23. MYSZKOWSKI TRANSPOSITION CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 3

##### HOME.2.23.3. DECRYPTION: MYSZKOWSKI TRANSPOSITION CIPHER

Enter The Cipher Text (A String Of Length Greater Than 2):  
TTnsa!ieThh-s e i!lIt Px

Enter The Keyword (A String Of Length Greater Than 1 And Less Than 100 And Less Than Length Of Cipher Text):

Computer

The Plain Text Is:

'This Is The Plain-Text!!'

#### HOME.2.23. MYSZKOWSKI TRANSPOSITION CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 4

#### HOME.2. CIPHERS

1. Rotation Cipher
  2. Atbash Cipher
  3. Caesar Positive Shift Cipher
  4. Caesar Negative Shift Cipher
  5. One Time Pad Cipher
  6. Monoalphabetic Substitution Cipher
  7. Polyalphabetic Substitution Cipher
  8. Autokey Cipher
  9. Running Key Cipher
  10. Polybius Square Cipher
  11. Bifid Cipher
  12. Trifid Cipher
  13. Bitwise Rotation Cipher
  14. Bitwise Atbash Cipher
  15. Bitwise Binary Complement Cipher
  16. Bitwise Left Shift Cipher
  17. Bitwise Right Shift Cipher
  18. Bitwise Exclusive Or Cipher
  19. Bitwise One Time Pad Cipher
  20. Square Transposition Cipher
  21. Columnar Transposition Cipher
  22. Double Transposition Cipher
  23. Myszowski Transposition Cipher
  24. Chinese Transposition Cipher
  25. Rail Fence Cipher
  26. Back To HOME
- Enter Choice: 24

#### HOME.2.24. CHINESE TRANSPOSITION CIPHER

1. About
  2. Encrypt
  3. Decrypt
  4. Back To CIPHERS
- Enter Choice: 1

#### HOME.2.24.1. ABOUT: CHINESE TRANSPOSITION CIPHER

In the Chinese Transposition Cipher, the letters of the message are written from right to left, down and up columns to scramble the letters. Then, starting in the first row, the letters are taken in order to get the new Cipher Text.

The Keyword determined the number of rows. All remaining blanks are filled with spaces.

#### HOME.2.24. CHINESE TRANSPOSITION CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 2

#### HOME.2.24.2. ENCRYPTION: CHINESE TRANSPOSITION CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):  
This Is The Plain-Text!!

Enter The Key (A Number Greater Than 1 And Less Than 100 And Less Than  
Length Of Plain Text):  
3

The Cipher Text Is:  
'!T-P sIT!enle htxiahTsi'

#### HOME.2.24. CHINESE TRANSPOSITION CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 3

#### HOME.2.24.3. DECRYPTION: CHINESE TRANSPOSITION CIPHER

Enter The Cipher Text (A String Of Length Greater Than 2):  
'!T-P sIT!enle htxiahTsi'

Enter The Key (A Number Greater Than 1 And Less Than 100 And Less Than  
Length Of Cipher Text):  
3

The Plain Text Is:  
'This Is The Plain-Text!!'

#### HOME.2.24. CHINESE TRANSPOSITION CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 4

#### HOME.2. CIPHERS

1. Rotation Cipher
2. Atbash Cipher
3. Caesar Positive Shift Cipher
4. Caesar Negative Shift Cipher
5. One Time Pad Cipher
6. Monoalphabetic Substitution Cipher

7. Polyalphabetic Substitution Cipher
8. Autokey Cipher
9. Running Key Cipher
10. Polybius Square Cipher
11. Bifid Cipher
12. Trifid Cipher
13. Bitwise Rotation Cipher
14. Bitwise Atbash Cipher
15. Bitwise Binary Complement Cipher
16. Bitwise Left Shift Cipher
17. Bitwise Right Shift Cipher
18. Bitwise Exclusive Or Cipher
19. Bitwise One Time Pad Cipher
20. Square Transposition Cipher
21. Columnar Transposition Cipher
22. Double Transposition Cipher
23. Myszkowski Transposition Cipher
24. Chinese Transposition Cipher
25. Rail Fence Cipher
26. Back To HOME

Enter Choice: 25

#### HOME.2.25. RAIL FENCE CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 1

#### HOME.2.25.1. ABOUT: RAIL FENCE CIPHER

The Rail Fence Cipher is a form of transposition cipher that gets its name from the way in which it is encoded. In the rail fence cipher, the Plain Text is written downwards on successive 'rails' of an imaginary fence, then moving up when we get to the bottom. Blanks are filled with spaces. The message is then read off in rows from left to right.

#### HOME.2.25. RAIL FENCE CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 2

#### HOME.2.25.2. ENCRYPTION: RAIL FENCE CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):  
This Is The Plain-Text!!

Enter The Key (A Number Greater Than 1 And Less Than The Length Of The Plain Text):

3

The Cipher Text Is:

'T TPnx hsI h li-et! iseaT! '

#### HOME.2.25. RAIL FENCE CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 3

#### HOME.2.25.3. DECRYPTION: RAIL FENCE CIPHER

Enter The Cipher Text (A String Of Length Greater Than 2):

T TPnx hsI h li-et! iseaT!

Enter The Key (A Number Greater Than 1 And Less Than The Length Of The Cipher Text):

3

The Plain Text Is:

'This Is The Plain-Text!!'

#### HOME.2.25. RAIL FENCE CIPHER

1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

Enter Choice: 4

#### HOME.2. CIPHERS

1. Rotation Cipher
2. Atbash Cipher
3. Caesar Positive Shift Cipher
4. Caesar Negative Shift Cipher
5. One Time Pad Cipher
6. Monoalphabetic Substitution Cipher
7. Polyalphabetic Substitution Cipher
8. Autokey Cipher
9. Running Key Cipher
10. Polybius Square Cipher
11. Bifid Cipher
12. Trifid Cipher
13. Bitwise Rotation Cipher
14. Bitwise Atbash Cipher
15. Bitwise Binary Compliment Cipher
16. Bitwise Left Shift Cipher
17. Bitwise Right Shift Cipher
18. Bitwise Exclusive Or Cipher
19. Bitwise One Time Pad Cipher
20. Square Transposition Cipher
21. Columnar Transposition Cipher



- 22. Double Transposition Cipher
  - 23. Myszowski Transposition Cipher
  - 24. Chinese Transposition Cipher
  - 25. Rail Fence Cipher
  - 26. Back To HOME
- Enter Choice: 26

HOME: IMPLEMENTATION OF CRYPTOGRAPHY IN JAVA

- 1. About
  - 2. Ciphers
  - 3. Author
  - 4. Exit
- Enter Choice: 3

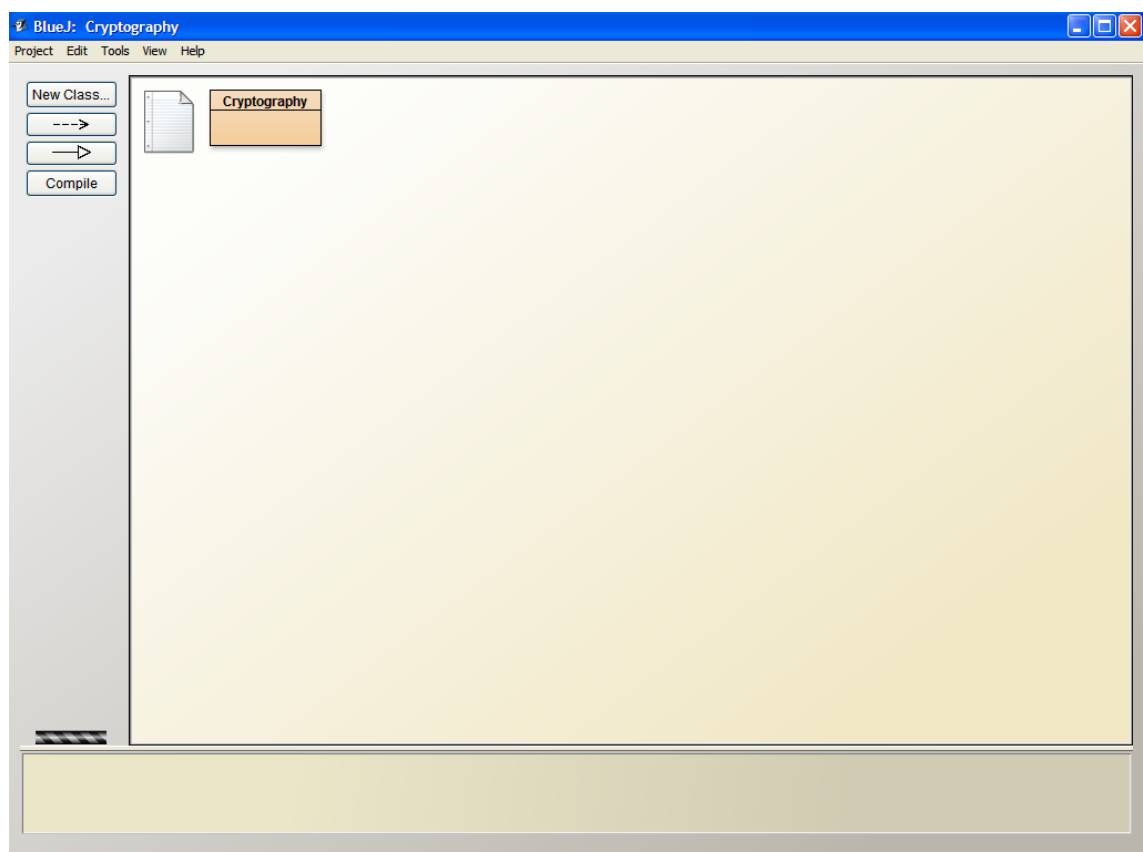
HOME.3. AUTHOR  
Name: Omkar S. Nath  
Class: 10–D  
Roll Number: 35  
Computer Code: 230438

HOME: IMPLEMENTATION OF CRYPTOGRAPHY IN JAVA

- 1. About
  - 2. Ciphers
  - 3. Author
  - 4. Exit
- Enter Choice: 4

Thank You For Using This Program.  
- Omkar S. Nath

# SCREENSHOTS



```

/*
 * ICSE Std. 10 Computer Science Project (Java)
 *
 * Title: Implementation Of Cryptography In Java
 *
 * Number of Lines of Code: 10,160
 *
 *
 * Summary:
 *
 * This Program Is A Menu Driven Program Demonstrating
 * The Implementation of Various Ciphers In Java. This
 * Program Also Includes Code For Validating The Input
 * Format Received From The User.
 *
 * The Following Ciphers Have Been Implemented In Java
 *
 * 1. Rotation Cipher
 * 2. Atbash Cipher
 * 3. Caesar Positive Shift Cipher
 * 4. Caesar Negative Shift Cipher
 * 5. One Time Pad Cipher
 * 6. Monoalphabetic Substitution Cipher
 * 7. Polyalphabetic Substitution Cipher
 * 8. Autokey Cipher
 * 9. Running Key Cipher
 * 10. Polybius Square Cipher
 * 11. Bifid Cipher
 * 12. Trifid Cipher
 * 13. Bitwise Rotation Cipher
 * 14. Bitwise Atbash Cipher
 * 15. Bitwise Binary Complement Cipher
 * 16. Bitwise Left Shift Cipher
 * 17. Bitwise Right Shift Cipher
 * 18. Bitwise Exclusive Or Cipher

```

```

import java.io.*;

class Cryptography
{
    public static int indent=1;

    public static void main()throws IOException
    {
        BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

        int f1;
        int f2;
        int f3;
        int ch1;
        int ch2;
        int ch3;
        int temp;

        ch1=0;
        f1=0;
        do
        {
            temp=0;
            while(temp==0)
            {
                System.out.println();
                System.out.println("HOME: IMPLEMENTATION OF CRYPTOGRAPHY IN JAVA");
                System.out.println("1. About");
                System.out.println("2. Ciphers");
                System.out.println("3. Author");
                System.out.println("4. Exit");
                System.out.print("Enter Choice: ");

                try
                {
                    ch1=Integer.parseInt(bfrd.readLine());

```

```

switch(ch1)
{
    case 1:
        f1=0;
        if(indent==1)System.out.print("\t");
        System.out.println();
        if(indent==1)System.out.print("\t");
        System.out.println("HOME.1. ABOUT: IMPLEMENTATION OF CRYPTOGRAPHY IN JAVA");
        if(indent==1)System.out.print("\t");
        System.out.println();
        if(indent==1)System.out.print("\t");
        System.out.println("Cryptography is the practice and study of techniques");
        if(indent==1)System.out.print("\t");
        System.out.println("for secure communication in the presence of third");
        if(indent==1)System.out.print("\t");
        System.out.println("parties (called adversaries). This program demonstrates");
        if(indent==1)System.out.print("\t");
        System.out.println("various ciphers in java. It accepts the Plain Text and");
        if(indent==1)System.out.print("\t");
        System.out.println("Key (wherever applicable) from the user and generates");
        if(indent==1)System.out.print("\t");
        System.out.println("the Cipher Text using various Encryption algorithms. The ");
        if(indent==1)System.out.print("\t");
        System.out.println("Plain Text is then obtained from the Cipher Text using");
        if(indent==1)System.out.print("\t");
        System.out.println("the same Key by applying the respective Decryption");
        if(indent==1)System.out.print("\t");
        System.out.println("algorithm.");
        break;

    case 2:
        f1=0;

        ch2=0;
        f2=0;
        do

```

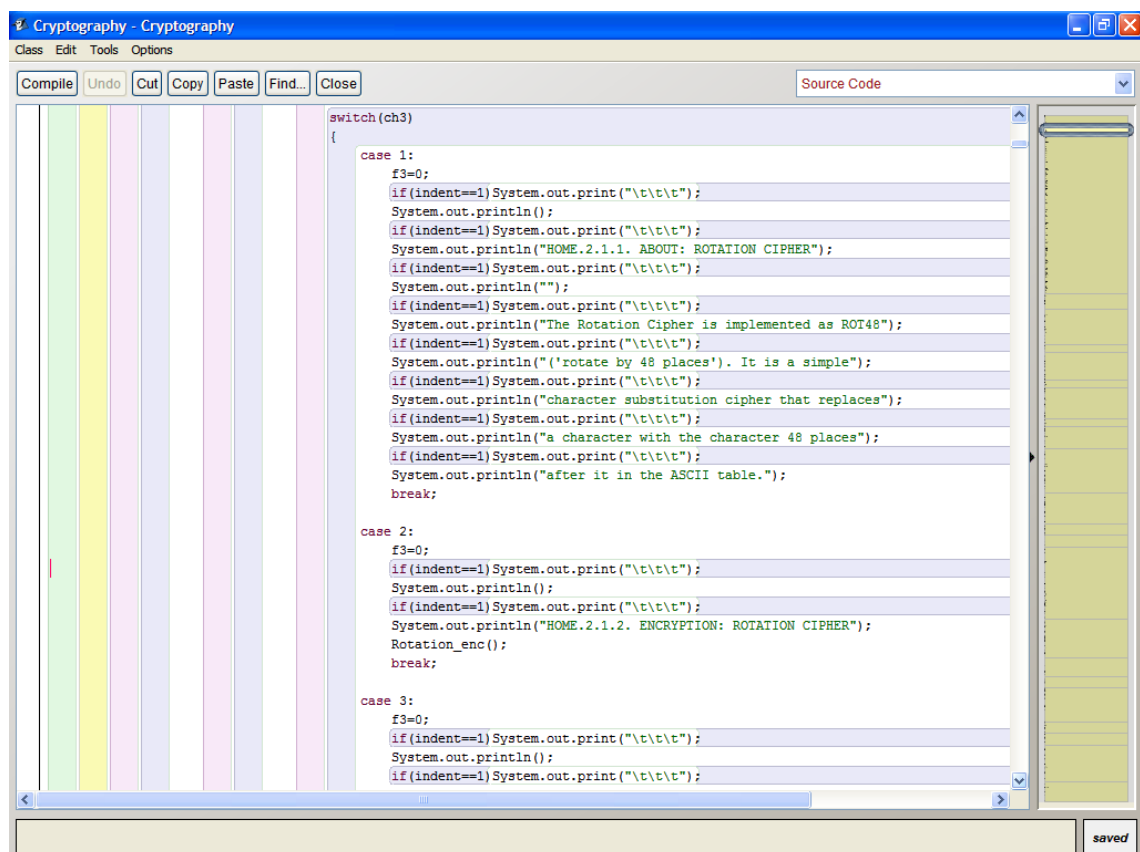
```

switch(ch2)
{
    case 1:
        f2=0;

        ch3=0;
        f3=0;
        do
        {
            temp=0;
            while(temp==0)
            {
                if(indent==1)System.out.print("\t\t");
                System.out.println();
                if(indent==1)System.out.print("\t\t");
                System.out.println("HOME.2.1. ROTATION CIPHER");
                if(indent==1)System.out.print("\t\t");
                System.out.println("1. About");
                if(indent==1)System.out.print("\t\t");
                System.out.println("2. Encrypt");
                if(indent==1)System.out.print("\t\t");
                System.out.println("3. Decrypt");
                if(indent==1)System.out.print("\t\t");
                System.out.println("4. Back To CIPHERS");
                if(indent==1)System.out.print("\t\t");
                System.out.print("Enter Choice: ");

                try
                {
                    ch3=Integer.parseInt(bf.readLine());
                    temp=1;
                }
                catch(NumberFormatException E1nfe)
                {
                    if(indent==1)System.out.print("\t\t");
                    System.out.println("Choice Should Be A Number");
                    temp=0;
                }
            }
        }

```



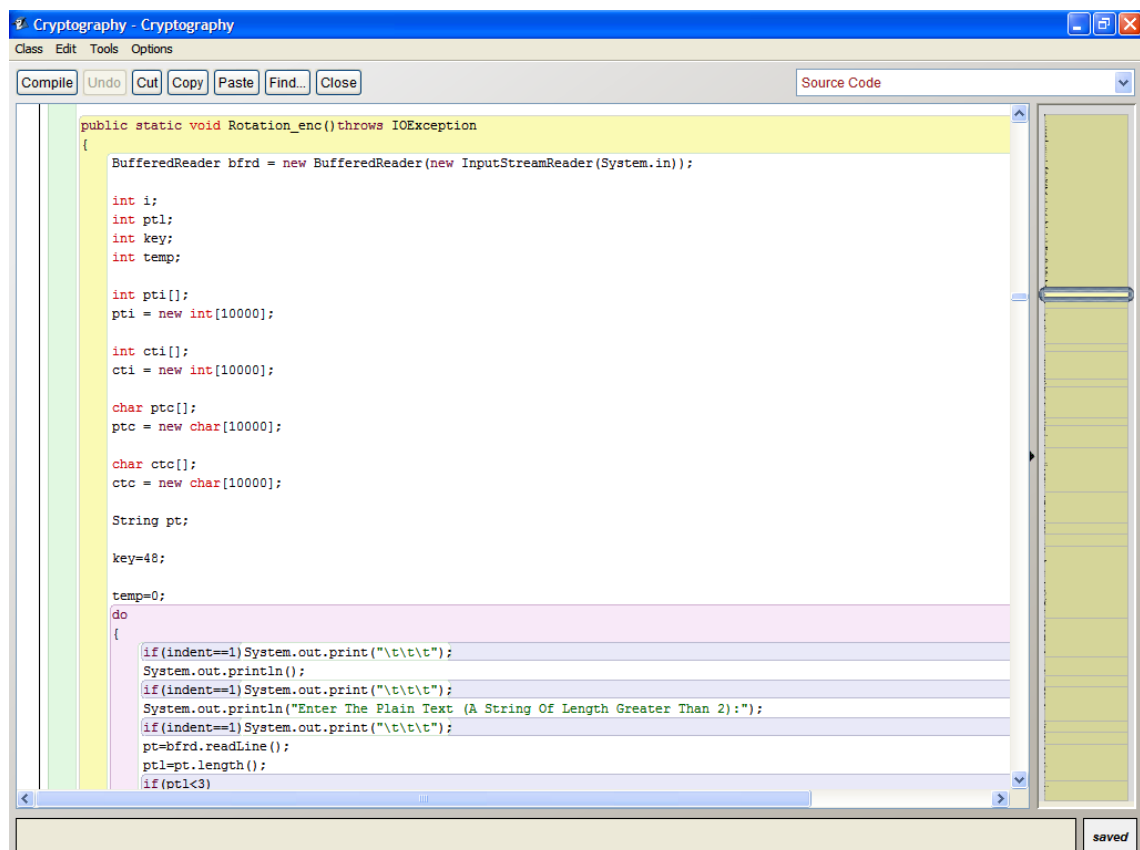
```

switch(ch3)
{
    case 1:
        f3=0;
        if(indent==1) System.out.print("\t\t\t\t");
        System.out.println();
        if(indent==1) System.out.print("\t\t\t\t");
        System.out.println("HOME.2.1.1. ABOUT: ROTATION CIPHER");
        if(indent==1) System.out.print("\t\t\t\t");
        System.out.println("");
        if(indent==1) System.out.print("\t\t\t\t");
        System.out.println("The Rotation Cipher is implemented as ROT48");
        if(indent==1) System.out.print("\t\t\t\t");
        System.out.println('rotate by 48 places'. It is a simple");
        if(indent==1) System.out.print("\t\t\t\t");
        System.out.println("character substitution cipher that replaces");
        if(indent==1) System.out.print("\t\t\t\t");
        System.out.println("a character with the character 48 places");
        if(indent==1) System.out.print("\t\t\t\t");
        System.out.println("after it in the ASCII table.");
        break;

    case 2:
        f3=0;
        if(indent==1) System.out.print("\t\t\t\t");
        System.out.println();
        if(indent==1) System.out.print("\t\t\t\t");
        System.out.println("HOME.2.1.2. ENCRYPTION: ROTATION CIPHER");
        Rotation_enc();
        break;

    case 3:
        f3=0;
        if(indent==1) System.out.print("\t\t\t\t");
        System.out.println();
        if(indent==1) System.out.print("\t\t\t\t");

```



```

public static void Rotation_enc()throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int ptl;
    int key;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    char ptc[];
    ptc = new char[10000];

    char ctc[];
    ctc = new char[10000];

    String pt;

    key=48;

    temp=0;
    do
    {
        if(indent==1) System.out.print("\t\t\t\t");
        System.out.println();
        if(indent==1) System.out.print("\t\t\t\t");
        System.out.println("Enter The Plain Text (A String Of Length Greater Than 2):");
        if(indent==1) System.out.print("\t\t\t\t");
        pt=bfrd.readLine();
        ptl=pt.length();
        if(ptl<3)

```

```

public static void Rotation_dec() throws IOException
{
    BufferedReader bfrd = new BufferedReader(new InputStreamReader(System.in));

    int i;
    int cti;
    int key;
    int temp;

    int pti[];
    pti = new int[10000];

    int cti[];
    cti = new int[10000];

    char ptc[];
    ptc = new char[10000];

    char ctc[];
    ctc = new char[10000];

    String ct;

    key=48;

    temp=0;
    do
    {
        if(indent==1) System.out.print("\t\t\t");
        System.out.println();
        if(indent==1) System.out.print("\t\t\t");
        System.out.println("Enter The Cipher Text (A String Of Length Greater Than 2):");
        if(indent==1) System.out.print("\t\t\t");
        ct=bfrd.readLine();
        cti=ct.length();
        if(ctl<3)

```

```

        {
            o=o+m;
            w=w-1;
            pti[j]=cti[w];
            j=j+1;
        }
    }

    for(i=0;i<cti;i++)
    {
        if(pti[i]!=32)
        {
            j=i;
        }
    }

    pti=j;

    if(indent==1) System.out.print("\t\t\t");
    System.out.println();
    if(indent==1) System.out.print("\t\t\t");
    System.out.println("The Plain Text Is:");
    if(indent==1) System.out.print("\t\t\t");
    System.out.print("");
    for(i=0;i<ptl;i++)
    {
        ptc[i]=(char)pti[i];
        System.out.print(ptc[i]);
    }
    System.out.println("");
}

//***** End Of Program *****/

```

```

BlueJ: Terminal Window - Cryptography
Options

HOME: IMPLEMENTATION OF CRYPTOGRAPHY IN JAVA
1. About
2. Ciphers
3. Author
4. Exit
Enter Choice: 1

HOME.1. ABOUT: IMPLEMENTATION OF CRYPTOGRAPHY IN JAVA

Cryptography is the practice and study of techniques
for secure communication in the presence of third
parties (called adversaries). This program demonstrates
various ciphers in java. It accepts the Plain Text and
Key (wherever applicable) from the user and generates
the Cipher Text using various Encryption algorithms. The
Plain Text is then obtained from the Cipher Text using
the same Key by applying the respective Decryption
algorithm.

HOME: IMPLEMENTATION OF CRYPTOGRAPHY IN JAVA
1. About
2. Ciphers
3. Author
4. Exit
Enter Choice: 3

HOME.3. AUTHOR
Name: Omkar S. Nath
Class: 10-D
Roll Number: 35
Computer Code: 230438

HOME: IMPLEMENTATION OF CRYPTOGRAPHY IN JAVA
1. About
2. Ciphers
3. Author
4. Exit
Enter Choice: 2

HOME.2. CIPHERS
1. Rotation Cipher

```

```

BlueJ: Terminal Window - Cryptography
Options

HOME: IMPLEMENTATION OF CRYPTOGRAPHY IN JAVA
1. About
2. Ciphers
3. Author
4. Exit
Enter Choice: 2

HOME.2. CIPHERS
1. Rotation Cipher
2. Atbash Cipher
3. Caesar Positive Shift Cipher
4. Caesar Negative Shift Cipher
5. One Time Pad Cipher
6. Monoalphabetic Substitution Cipher
7. Polyalphabetic Substitution Cipher
8. Autokey Cipher
9. Running Key Cipher
10. Polybius Square Cipher
11. Bifid Cipher
12. Trifid Cipher
13. Bitwise Rotation Cipher
14. Bitwise Atbash Cipher
15. Bitwise Binary Complement Cipher
16. Bitwise Left Shift Cipher
17. Bitwise Right Shift Cipher
18. Bitwise Exclusive Or Cipher
19. Bitwise One Time Pad Cipher
20. Square Transposition Cipher
21. Columnar Transposition Cipher
22. Double Transposition Cipher
23. Myszkowski Transposition Cipher
24. Chinese Transposition Cipher
25. Rail Fence Cipher
26. Back To HOME
Enter Choice: 1

HOME.2.1. ROTATION CIPHER
1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS
Enter Choice: 1

```

```

BlueJ: Terminal Window - Cryptography
Options
26. BACK TO HOME
Enter Choice: 1

HOME.2.1. ROTATION CIPHER
1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS
Enter Choice: 1

HOME.2.1.1. ABOUT: ROTATION CIPHER

The Rotation Cipher is implemented as ROT48
('rotate by 48 places'). It is a simple
character substitution cipher that replaces
a character with the character 48 places
after it in the ASCII table.

HOME.2.1. ROTATION CIPHER
1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS
Enter Choice: 2

HOME.2.1.2. ENCRYPTION: ROTATION CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):
This Is The Plain-Text!!

The Cipher Text Is:
'%9:DPyDP%96P!=2:~}%6IEQQ'

HOME.2.1. ROTATION CIPHER
1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS
Enter Choice: 3

HOME.2.1.3. DECRYPTION: ROTATION CIPHER

Enter The Cipher Text (A String Of Length Greater Than 2):

```

```

BlueJ: Terminal Window - Cryptography
Options
Enter Choice: 3

HOME.2.1.3. DECRYPTION: ROTATION CIPHER

Enter The Cipher Text (A String Of Length Greater Than 2):
%9:DPyDP%96P!=2:~}%6IEQQ

The Plain Text Is:
'This Is The Plain-Text!!'

HOME.2.1. ROTATION CIPHER
1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS
Enter Choice: 4

HOME.2. CIPHERS
1. Rotation Cipher
2. Atbash Cipher
3. Caesar Positive Shift Cipher
4. Caesar Negative Shift Cipher
5. One Time Pad Cipher
6. Monoalphabetic Substitution Cipher
7. Polyalphabetic Substitution Cipher
8. Autokey Cipher
9. Running Key Cipher
10. Polybius Square Cipher
11. Bifid Cipher
12. Trifid Cipher
13. Bitwise Rotation Cipher
14. Bitwise Atbash Cipher
15. Bitwise Binary Complement Cipher
16. Bitwise Left Shift Cipher
17. Bitwise Right Shift Cipher
18. Bitwise Exclusive Or Cipher
19. Bitwise One Time Pad Cipher
20. Square Transposition Cipher
21. Columnar Transposition Cipher
22. Double Transposition Cipher
23. Myszkowski Transposition Cipher
24. Chinese Transposition Cipher

```



```

BlueJ: Terminal Window - Cryptography
Options
25. Rail Fence Cipher
26. Back To HOME
Enter Choice: 2

HOME.2.2. ATBASH CIPHER
1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS
Enter Choice: 1

HOME.2.2.1. ABOUT: ATBASH CIPHER

Atbash is a simple substitution cipher
for the Hebrew alphabet. It consists
of substituting aleph (the first letter)
for tav (the last), beth (the second)
for shin (one before last), and so on,
reversing the alphabet. The same
algorithm is applied to the ASCII
character.

HOME.2.2.2. ATBASH CIPHER
1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS
Enter Choice: 2

HOME.2.2.2. ENCRYPTION: ATBASH CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):
This Is The Plain-Text!!

The Cipher Text Is:
'J65+-U+-J69-N2=50qJ9s*}'

HOME.2.2. ATBASH CIPHER
1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS

```

```

BlueJ: Terminal Window - Cryptography
Options

The Cipher Text Is:
'J65+-U+-J69-N2=50qJ9s*}'

HOME.2.2. ATBASH CIPHER
1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS
Enter Choice: 3

HOME.2.2.3. DECRYPTION: ATBASH CIPHER

Enter The Cipher Text (A String Of Length Greater Than 2):
J65+-U+-J69-N2=50qJ9s*}

The Plain Text Is:
'This Is The Plain-Text!!'

HOME.2.2. ATBASH CIPHER
1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS
Enter Choice: 4

HOME.2. CIPHERS
1. Rotation Cipher
2. Atbash Cipher
3. Caesar Positive Shift Cipher
4. Caesar Negative Shift Cipher
5. One Time Pad Cipher
6. Monoalphabetic Substitution Cipher
7. Polyalphabetic Substitution Cipher
8. Autokey Cipher
9. Running Key Cipher
10. Polybius Square Cipher
11. Bifid Cipher
12. Trifid Cipher
13. Bitwise Rotation Cipher
14. Bitwise Atbash Cipher
15. Bitwise Binary Complement Cipher
16. Bitwise Left Shift Cipher

```

```

BlueJ: Terminal Window - Cryptography
Options
24. Chinese Transposition Cipher
25. Rail Fence Cipher
26. Back To HOME
Enter Choice: 3

HOME.2.3. CAESAR POSITIVE SHIFT CIPHER
1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS
Enter Choice: 1

HOME.2.3.1. ABOUT: CAESAR POSITIVE SHIFT CIPHER

The Caesar Positive Shift Cipher is implemented
as a rotation cipher. It is a simple character
substitution cipher that replaces a character
with the character 'Key' number of places after
it in the ASCII table. The value of 'Key' is taken
as input from the user.

HOME.2.3. CAESAR POSITIVE SHIFT CIPHER
1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS
Enter Choice: 2

HOME.2.3.2. ENCRYPTION: CAESAR POSITIVE SHIFT CIPHER

Enter The Plain Text (A String Of Length Greater Than 2):
This Is The Plain-Text!!

Enter the Key (A Number Between 0 and 10000):
3

The Cipher Text Is:
'Wklv#Lv#Wkh#Sodlq0Wh{w6$'

HOME.2.3. CAESAR POSITIVE SHIFT CIPHER
1. About
2. Encrypt
3. Decrypt

```

```

BlueJ: Terminal Window - Cryptography
Options

HOME.2.3. CAESAR POSITIVE SHIFT CIPHER
1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS
Enter Choice: 3

HOME.2.3.3. DECRYPTION: CAESAR POSITIVE SHIFT CIPHER

Enter The Cipher Text (A String Of Length Greater Than 2):
Wklv#Lv#Wkh#Sodlq0Wh{w6$

Enter the Key (A Number Between 0 and 10000):
3

The Plain Text Is:
'This Is The Plain-Text!!'

HOME.2.3. CAESAR POSITIVE SHIFT CIPHER
1. About
2. Encrypt
3. Decrypt
4. Back To CIPHERS
Enter Choice: 4

HOME.2. CIPHERS
1. Rotation Cipher
2. Atbash Cipher
3. Caesar Positive Shift Cipher
4. Caesar Negative Shift Cipher
5. One Time Pad Cipher
6. Monoalphabetic Substitution Cipher
7. Polyalphabetic Substitution Cipher
8. Autokey Cipher
9. Running Key Cipher
10. Polybius Square Cipher
11. Bifid Cipher
12. Trifid Cipher
13. Bitwise Rotation Cipher
14. Bitwise Atbash Cipher
15. Bitwise Binary Complement Cipher

```

```

BlueJ: Terminal Window - Cryptography
Options
    3. Decrypt
    4. Back To CIPHERS
    Enter Choice: 2

    HOME.2.25.2. ENCRYPTION: RAIL FENCE CIPHER

    Enter The Plain Text (A String Of Length Greater Than 2):
    This Is The Plain-Text!!

    Enter The Key (A Number Greater Than 1 And Less Than The Length Of The Plain Text):
    3

    The Cipher Text Is:
    'T TPNx hsI h li-et! iseaT! '

    HOME.2.25. RAIL FENCE CIPHER
    1. About
    2. Encrypt
    3. Decrypt
    4. Back To CIPHERS
    Enter Choice: 3

    HOME.2.25.3. DECRYPTION: RAIL FENCE CIPHER

    Enter The Cipher Text (A String Of Length Greater Than 2):
    T TPNx hsI h li-et! iseaT!

    Enter The Key (A Number Greater Than 1 And Less Than The Length Of The Cipher Text):
    3

    The Plain Text Is:
    'This Is The Plain-Text!!'

    HOME.2.25. RAIL FENCE CIPHER
    1. About
    2. Encrypt
    3. Decrypt
    4. Back To CIPHERS
    Enter Choice: 4

    HOME.2. CIPHERS
    1. Rotation Cipher
  
```

```

BlueJ: Terminal Window - Cryptography
Options
    10. Polybius Square Cipher
    11. Bifid Cipher
    12. Trifid Cipher
    13. Bitwise Rotation Cipher
    14. Bitwise Atbash Cipher
    15. Bitwise Binary Complement Cipher
    16. Bitwise Left Shift Cipher
    17. Bitwise Right Shift Cipher
    18. Bitwise Exclusive Or Cipher
    19. Bitwise One Time Pad Cipher
    20. Square Transposition Cipher
    21. Columnar Transposition Cipher
    22. Double Transposition Cipher
    23. Myszowski Transposition Cipher
    24. Chinese Transposition Cipher
    25. Rail Fence Cipher
    26. Back To HOME
    Enter Choice: 26

    HOME: IMPLEMENTATION OF CRYPTOGRAPHY IN JAVA
    1. About
    2. Ciphers
    3. Author
    4. Exit
    Enter Choice: 3

    HOME.3. AUTHOR
    Name: Omkar S. Nath
    Class: 10-D
    Roll Number: 35
    Computer Code: 230438

    HOME: IMPLEMENTATION OF CRYPTOGRAPHY IN JAVA
    1. About
    2. Ciphers
    3. Author
    4. Exit
    Enter Choice: 4

    Thank You For Using This Program.
    - Omkar S. Nath
  
```

# CONCLUSION

This program demonstrates various ciphers in java. It accepts the Plain Text and Key (wherever applicable) from the user and generates the Cipher Text using various Encryption algorithms. The Plain Text is then obtained from the Cipher Text using the same Key by applying the respective Decryption algorithm.

The developed program can be used to encrypt and decrypt strings of characters. Depending on the Cipher used a key may or may not be needed. The Plain Text is readable but the Cipher Text is not understandable without knowledge of the algorithm and the key used for encryption. The Cipher Text can thus be then sent over any communication media where, even if it is intercepted, it will be difficult for the eavesdropper to interpret and understand the message.

All the Ciphers in the program are Symmetric Key Ciphers. This means that the same key is used for both the processes of encryption and decryption. Thus the sender and the receiver both need to have the knowledge of what algorithm is being used along with the key.

The strength of a Cipher can be increased by nesting the Ciphers. This means that the output of one Cipher is given as input to another or the same Cipher. At each stage the same or a different key can be used. This can be done in any permutation as many times as the user wishes to do so. To obtain the original text one needs to apply the Ciphers in the reverse order that it was applied while encrypting the raw plain text data.

The main purpose of Cryptographical Ciphers is to maintain the privacy of data. This means that only people who are authorized to have access to the data can view the data. This prevents others who may eavesdrop for gain unauthorized access to the data from understanding the data. The ability to maintain secrets is a basic need for humans. In today's electronic age, one can employ the techniques implemented in this project to fulfill this need.

# **FUTURE ENHANCEMENT**

1. The codes in this project can be used to secure email and other electronic communications.
2. These cryptographic algorithms can be applied to files such as text, image and video files.
3. Complex cryptographic algorithms, such as the Data Encryption Standard (DES) and the Advanced Encryption Standard (AES) can be implemented.
4. The key generation for the symmetric key cryptographic algorithm can be automated and randomized.
5. The User Interface (UI) can be made graphical in nature to make the program easy to use.

# BIBLIOGRAPHY

1. 'Computer Applications A Text Book for Class X' By Sumita Arora.
2. 'Cryptography and Network Security' By William Stallings.
3. Encyclopedia Britannica.
4. Encyclopedia Encarta.
5. Encyclopedia Wikipedia.
6. 'Frank Computer Applications For ICSE Class IX' By Sonia Sabharwal.
7. 'Frank Computer Applications For ICSE Class X' By Sonia Sabharwal.
8. 'Programming with Java A Primer' By E Balagurusamy.

# CD CONTAINING PROGRAM

*This page intentionally left blank.*