Advanced Security 1 – DT211-4, DT282-4 and DT228-4

Lab Sheet 3 (2 Mark)Stephen Darcy – C18490924

Part A

*package* lab3;  
  
*/\*  
Author: Stephen Darcy  
Date: 10/11/2021  
Description: Write a Java program (or any other programming language you are happy to use) to encrypt  
plaintext using a 2 \* 2 Hill cipher.  
\*/  
  
import* java.util.Scanner;  
  
*public class* Main {  
 *static* Scanner scanner = *new* Scanner(System.in);  
  
 *public static void* main(String[] args) {  
 *//get user plaintext to be encrypted* System.out.println("Hill Cipher 2\*2 Matrix Encryption");  
 System.out.println("Enter plaintext to encrypt: ");  
 String plaintext = scanner.nextLine();  
  
 *//get key to encrypt plaintext with* System.out.println("Enter a 4 letter key:");  
 String key = scanner.nextLine();  
  
 *//make all uppercase and remove whitespace with regex* plaintext = plaintext.replaceAll("\\s", "").toUpperCase();  
 key = key.replaceAll("\\s", "").toUpperCase();  
  
 *//making plaintext length an even number (adding z to end if odd)  
 if* (plaintext.length() % 2 != 0) {  
 plaintext += "Z";  
 }  
  
 *//creating 2 2D arrays to hold the plaintext matrix and key matrix  
 int*[][] plaintextMatrix = *new int*[2][plaintext.length()];  
 *int*[][] keyMatrix = *new int*[2][2];  
  
 *//using a for loop convert the plaintext to numbers and insert into matrix  
 //using 65 as this is the ascii code for 'A' and plaintext is all uppercase  
 int* row1Index = 0, row2Index = 0;  
 *for* (*int* i = 0; i < plaintext.length(); i++) {  
 *char* current = plaintext.charAt(i);  
 *if* (i % 2 == 0) {  
 plaintextMatrix[0][row1Index] = ((*int*) current) - 65;  
 row1Index++;  
 } *else* {  
 plaintextMatrix[1][row2Index] = ((*int*) current) - 65;  
 row2Index++;  
 }  
 }  
  
 *//same as above but with the key into a 2\*2 matrix  
 int* rowIndex = 0;  
 *for* (*int* i = 0; i < 2; i++) {  
 *for* (*int* j = 0; j < 2; j++) {  
 *char* current = key.charAt(rowIndex);  
 keyMatrix[i][j] = ((*int*) current) - 65;  
 rowIndex++;  
 }  
 }  
  
 *//finding the product of the key and plaintext pair mod 26 by doing matrix multiplication  
 //on the 2D arrays* StringBuilder ciphertext = *new* StringBuilder();  
 *int* currentRow1, currentRow2;  
 *for* (*int* i = 0; i < plaintext.length() / 2; i++) {  
 currentRow1 = ((plaintextMatrix[0][i] \* keyMatrix[0][0]) + (plaintextMatrix[1][i] \* keyMatrix[0][1])) % 26;  
 currentRow2 = ((plaintextMatrix[0][i] \* keyMatrix[1][0]) + (plaintextMatrix[1][i] \* keyMatrix[1][1])) % 26;  
  
 *//adding to stringbuilder* ciphertext.append((*char*) (currentRow1 + 65));  
 ciphertext.append((*char*) (currentRow2 + 65));  
 }  
  
 *//printing the ciphertext* System.out.println(ciphertext);  
 }  
}

Part B

*package* lab3;  
  
*/\*  
Author: Stephen Darcy  
Date: 10/11/2021  
Description: Write a Java program (or any other programming language you are happy to use) to perform a  
letter frequency attack on any monoalphabetic substitution cipher without human intervention.  
Your software should produce possible plaintexts in rough order of likelihood. It would be good  
if your user interface allowed the user to specify “give me the top 5 possible plaintexts.”  
\*/  
  
import* java.util.Scanner;  
  
*public class* Main {  
 *static* Scanner scanner = *new* Scanner(System.in);  
  
 *public static void* main(String[] args) {  
 *//get user plaintext to be encrypted* System.out.println("Letter frequency attack on any monoalphabetic substitution cipher");  
 System.out.println("Enter ciphertext to decrypt: ");  
 String ciphertext = scanner.nextLine();  
  
 *//get number of possible plaintexts to display* System.out.println("Enter the number of possible plaintexts to display: ");  
 *int* num = scanner.nextInt();  
  
 *//getting results* letterFrequencyAttack(ciphertext, num);  
 }  
  
 */\*\*  
 \* Function that uses the frequency of the letters in the english alphabet to decrypt a ciphertext  
 \* Frequency used - ETAOINSHRDLCUMWFGYPBVKJXQZ  
 \*  
 \* @param ciphertext the ciphertext entered by the user to be decrypted  
 \* @param num the amount of possible plaintexts to be displayed  
 \*/  
 private static void* letterFrequencyAttack(String ciphertext, *int* num) {  
 *//string for frequency of english letters from labsheet* String englishFrequency = "ETAOINSHRDLCUMWFGYPBVKJXQZ";  
  
 *//array to hold all the possible plaintexts* String[] plaintexts = *new* String[num];  
  
 *//make ciphertext uppercase and remove whitespace with regex* ciphertext = ciphertext.replaceAll("\\s", "").toUpperCase();  
  
 *//getting the frequency of each letter in the ciphertext  
 int* ciphertextFrequency[] = *new int*[26];  
 *for* (*int* i = 0; i < ciphertext.length(); i++) {  
 ciphertextFrequency[ciphertext.charAt(i) - 'A']++;  
 }  
  
 *//clone contents of ciphertextFrequency to new array  
 int* ciphertextFrequencyReverse[] = *new int*[26];  
 ciphertextFrequencyReverse = ciphertextFrequency.clone();  
  
 *//reverse contents of new array  
 int* arrayLength = ciphertextFrequencyReverse.length;  
 *for* (*int* i = 0; i < arrayLength / 2; i++) {  
 *int* current = ciphertextFrequencyReverse[i];  
 ciphertextFrequencyReverse[i] = ciphertextFrequencyReverse[arrayLength - i - 1];  
 ciphertextFrequencyReverse[arrayLength - i - 1] = current;  
 }  
  
 *//buffer array to see if current letter has been checked  
 int* buffer[] = *new int*[26];  
  
 *//start of decryption  
 for* (*int* i = 0; i < num; i++) {  
 *//finding the most common letter in ciphertext  
 int* mostCommon = 0;  
 *//iterating over array to see if letter is most common  
 for* (*int* j = 0; j < arrayLength; j++) {  
 *if* ((ciphertextFrequencyReverse[j] == ciphertextFrequency[i]) && buffer[j] == 0) {  
 *//setting buffer to any number but zero so won't be checked again* buffer[j] = 99;  
 mostCommon = j;  
 *break*;  
 }  
 }  
  
 *//getting the ascii for the most likely shift  
 int* shift = (englishFrequency.charAt(i) - 'A') - mostCommon;  
  
 *//current plaintext string* StringBuilder plaintext = *new* StringBuilder();  
  
 *//getting the number of plaintexts the user requested one letter at a time  
 for* (*int* j = 0; j < ciphertext.length(); j++) {  
 *//shifting each letter using modulus  
 int* current = ciphertext.charAt(j) - 'A';  
 current += shift;  
 *//making sure current is in range  
 if* (current < 0) {  
 current += 26;  
 }  
 *if* (current > 25) {  
 current -= 26;  
 }  
  
 *//converting ascii to letter and adding to plaintext* plaintext.append((*char*) ('A' + current));  
 }  
 plaintexts[i] = plaintext.toString();  
 }  
 *//output the plaintexts  
 for* (*int* i = 0; i < num; i++) {  
 System.out.println(plaintexts[i]);  
 }  
  
 }  
}