*#----------------------------------------------------------------------------------  
#Rutvik Parmar  
#H-41  
#Date - 8th Feb 2016  
#Assignement No:3 - PlayFair Cipher implementation for alphanumeric input  
#----------------------------------------------------------------------------------*alphanumeric = [**'A'**,**'B'**,**'C'**,**'D'**,**'E'**,**'F'**,  
 **'G'**,**'H'**,**'I'**,**'J'**,**'K'**,**'L'**,  
 **'M'**,**'N'**,**'O'**,**'P'**,**'Q'**,**'R'**,  
 **'S'**,**'T'**,**'U'**,**'V'**,**'W'**,**'X'**,  
 **'Y'**,**'Z'**,**'0'**,**'1'**,**'2'**,**'3'**,  
 **'4'**,**'5'**,**'6'**,**'7'**,**'8'**,**'9'**]  
  
**"""  
Function: key\_key()  
Brief: Receives the key as input from the user. Handles valid characters and  
 special cases.  
Parameter: None  
Return: Returns the key entered by the user.  
"""  
def** get\_key():  
 k = raw\_input().upper()  
 key = []  
 **for** char **in** k:  
 **if** char **in** alphanumeric **and** char **not in** key: *# add the character to the matrix if it's valid and not already in the matrix* key.append(char)  
 **elif** char **is "J"**: *# handle the case when the letter J appears in the key* key.append(**"I"**)  
 **for** char **in** alphanumeric:  
 **if** char **not in** key: *# add the rest of the alphahet not appearing in the key to the matrix* key.append(char)  
 **return** key  
  
  
**"""  
Function: gen\_matrix(key)  
Brief: Generates a Playfair matrix with the given key.  
Parameter: key - The key to use for generating a Playfair matrix.  
Return: Returns the keyed Playfair matrix.  
"""  
def** gen\_matrix(key):  
 matrix = []  
 counter = 0  
 **if** key == **''**: *# create a blank matrix* **for** xcounter **in** xrange(6):  
 x = []  
 **for** ycounter **in** xrange(6):  
 x.append(alphanumeric[counter])  
 counter += 1  
 matrix.append(x)  
 **else**: *# create a keyed matrix* **for** xcounter **in** xrange(6):  
 x = []  
 **for** ycounter **in** xrange(6):  
 x.append(key[counter])  
 counter += 1  
 matrix.append(x)  
 **return** matrix  
  
  
**"""  
Function: print\_matrix(matrix)  
Brief: Prints the given Playfair matrix.  
Parameter: matrix - The Playfair matrix to print out.  
Returns: None.  
"""  
def** print\_matrix(matrix):  
 **for** counter **in** xrange(6):  
 **print "%c %c %c %c %c %c"** % (matrix[counter][0], matrix[counter][1], matrix[counter][2], matrix[counter][3], matrix[counter][4],matrix[counter][5])  
 **print "\n"  
  
  
"""  
Fucntion: get\_message()  
Brief: Receives a message an input from the user. Handles valid characters and  
 special cases.  
Parameter: None  
Return: Returns the resulting message.  
"""  
def** get\_message():  
 m = raw\_input()  
 m2 = []  
 **for** char **in** m.upper():  
 **if** char **in** alphanumeric: *# handle valid characters in the message* m2.append(char)  
 **elif** char == **"J"**: *# handle the case when "J" appears in the message* m2.append(**"I"**)  
 **elif** char == **"."**: *# swap out the period with an x, for convenience* m2.append(**"X"**)  
 **return ''**.join(m2)  
  
  
**"""  
Function encrypt(message, key\_matrix)  
Brief: Performs encryption of the given message with the keyed Playfair  
 matrix.  
Parameter: message - The message to perform encryption on.  
Parameter: key\_matrix - The keyed Playfair matrix to use for encryption.  
Return: Returns nothing, the resulting ciphertext is printed at the end  
 of the function.  
"""  
def** encrypt(message, key\_matrix):  
 coords = []  
 ciphertext = []  
 digraphs = parse\_message(message)  
  
 **for** d **in** digraphs:  
 swap = []  
 temp = []  
 coords = get\_coords(d, key\_matrix)  
 **if** coords[0][0] == coords[1][0]: *# digraph lies on same x axis* x,y = ((coords[0][0], (coords[0][1] + 1) % 6))  
 swap.append((x,y))  
 x,y = ((coords[1][0], (coords[1][1] + 1) % 6))  
 swap.append((x,y))  
 **elif** coords[0][1] == coords[1][1]: *# digraph lies on same y axis* x,y = (((coords[0][0] + 1) % 6), coords[0][1])  
 swap.append((x,y))  
 x,y = (((coords[1][0] + 1) % 6), coords[1][1])  
 swap.append((x,y))  
 **else**: *# digraph lies on different x & y axis* swap.append((coords[0][0], coords[1][1]))  
 swap.append((coords[1][0], coords[0][1]))  
  
 **for** x,y **in** swap:  
 ciphertext.append(key\_matrix[x][y])  
  
  
 **print "Your encrypted message is: %s "** % **''**.join(ciphertext)  
  
  
  
**"""  
Function decrypt(message, key\_matrix)  
Brief: Performs decryption of the given message with the keyed Playfair  
 matrix.  
Parameter: message - The message to perform decryption on.  
Parameter: key\_matrix - The keyed Playfair matrix to use for decryption.  
Return: Returns nothing, the resulting plaintext is printed at the end  
 of the function.  
"""  
def** decrypt(message, key\_matrix):  
 coords = []  
 plaintext = []  
 digraphs = parse\_message(message)  
  
 **for** d **in** digraphs:  
 swap = []  
 temp = []  
 coords = get\_coords(d, key\_matrix)  
 **if** coords[0][0] == coords[1][0]: *# digraph lies on same x axis* x,y = ((coords[0][0], (coords[0][1] - 1) % 6))  
 swap.append((x,y))  
 x,y = ((coords[1][0], (coords[1][1] - 1) % 6))  
 swap.append((x,y))  
 **elif** coords[0][1] == coords[1][1]: *# digraph lies on same y axis* x,y = (((coords[0][0] - 1) % 6), coords[0][1])  
 swap.append((x,y))  
 x,y = (((coords[1][0] - 1) % 6), coords[1][1])  
 swap.append((x,y))  
 **else**: *# digraph lies on different x & y axis* swap.append((coords[0][0], coords[1][1]))  
 swap.append((coords[1][0], coords[0][1]))  
  
 **for** x,y **in** swap:  
 plaintext.append(key\_matrix[x][y])  
  
 **print "Your decrypted message is: %s "** % **''**.join(plaintext)  
  
  
**"""  
Function: parse\_message(message)  
Brief: Parses the message provided by the user. Prepares the text by handling  
 cases where double letters appear next to each other. Ignores non-alphabetic  
 characters, numbers, and punctuation.  
Parameter: message - The message entered by the user.  
Return: Returns an array of digraphs resulting from the given message.  
"""  
def** parse\_message(message):  
 digraphs = []  
 **while** len(message) > 0:  
 digraph = message[:2]  
 **if** len(digraph) == 1: *# trailing single chracter at the end of the message* digraph = digraph = **"%c%c"** % (digraph[0], **"X"**)  
 digraphs.append(digraph)  
 message = message[1:]  
 **elif** digraph[0] == digraph[1]: *# handle double letters appearing in the same digraph* digraph = **"%c%c"** % (digraph[0], **"X"**)  
 digraphs.append(digraph)  
 message = message[1:]  
 **else**: *# add the digraph to the list* digraphs.append(digraph)  
 message = message[2:]  
  
 **return** digraphs  
  
  
**"""  
Function: get\_coords(digraph, key\_matrix)  
Brief: Returns the coordinates of the letters in the given digraph from the provided keyed matrix.  
Parameter: digraph - The two-letter digraph to lookup in the key matrix.  
Parameter: key\_matrix - The keyed Playfair matrix to perform the lookup on.  
Return: Returns an array with the coordinates of the given digraph.  
"""  
def** get\_coords(digraph, key\_matrix):  
 coords = []  
 **for** char **in** digraph:  
 **for** x **in** xrange(6):  
 **for** y **in** xrange(6):  
 **if** key\_matrix[x][y] == char:  
 coords.append((x,y))  
 **return** coords  
  
  
**def** main():  
 m = gen\_matrix(**''**)  
 **print "\nInitial PLAYFAIR matrix:\n"** print\_matrix(m)  
  
 **print "Enter a key:"** k = get\_key()  
  
 **print "\nKeyed PLAYFAIR matrix:\n"** m = gen\_matrix(k)  
 print\_matrix(m)  
  
  
  
  
 **print "\nEncrypt Message:"  
 print "Enter the message you would like to encrypt."** message = get\_message()  
 **print "The message you entered was: %s"** % message  
 encrypt(message, m)  
  
  
  
  
 **print "\nDecrypt Message:"  
 print "Enter the message you would like to decrypt."** message = get\_message()  
 **print "The message you entered was: %s"** % message  
 decrypt(message, m)  
  
  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 main()

**'''  
Output -  
  
Initial PLAYFAIR matrix:  
  
A B C D E F  
G H I J K L  
M N O P Q R  
S T U V W X  
Y Z 0 1 2 3  
4 5 6 7 8 9  
  
  
Enter a key:  
encrypt123decrypt  
  
Keyed PLAYFAIR matrix:  
  
E N C R Y P  
T 1 2 3 D A  
B F G H I J  
K L M O Q S  
U V W X Z 0  
4 5 6 7 8 9  
  
  
  
Encrypt Message:  
Enter the message you would like to encrypt.  
rutvik789parmar  
The message you entered was: RUTVIK789PARMAR  
Your encrypted message is: EX1UBQ89PA3PS237  
  
Decrypt Message:  
Enter the message you would like to decrypt.  
EX1UBQ89PA3PS237  
The message you entered was: EX1UBQ89PA3PS237  
Your decrypted message is: RUTVIK789PARMARX  
  
'''**