Cryptography and Network Security
Lab Assignment - I

Program 1: Caesar Cipher

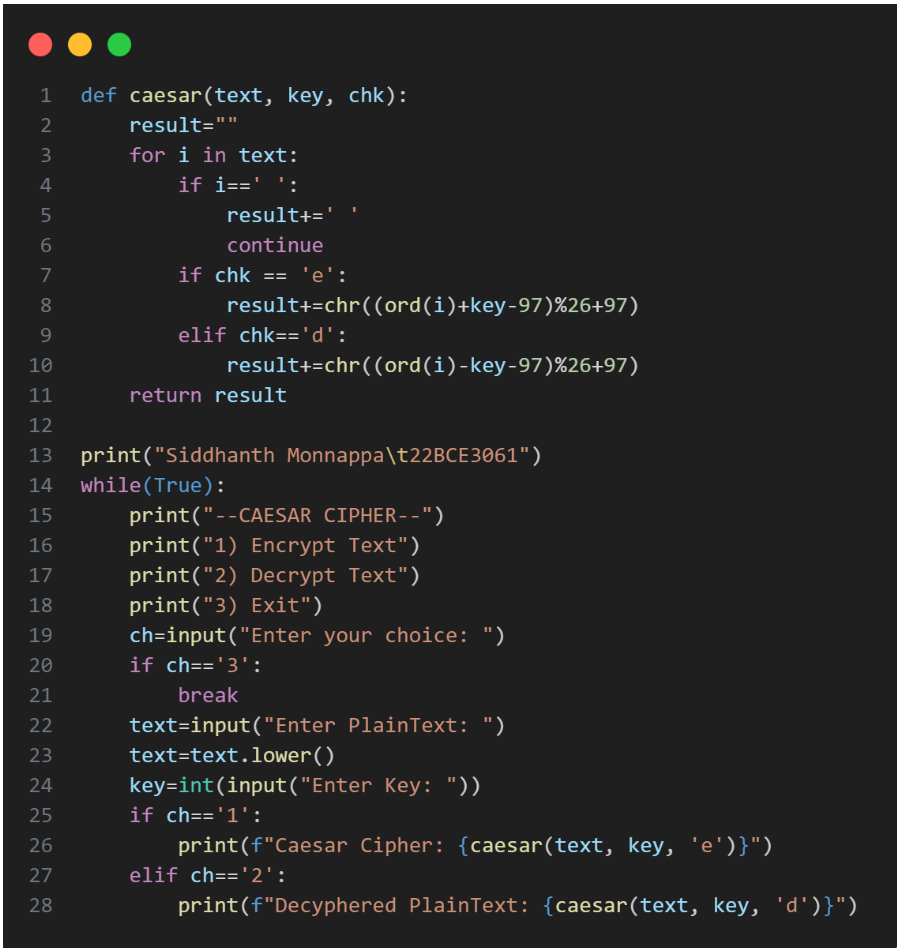
Pseudocode/Algorithm

- **Start**: Display the program details and options menu for encryption, decryption, or exit.
- **Input Choice**: Take the user’s choice for encryption, decryption, or exiting the program.
- **Exit Check**: If the choice is to exit, terminate the program.
- **Input Text and Key**: Prompt the user to enter the plaintext and the key.
- **Encrypt/Decrypt**: Based on the choice, call the caesar function with the appropriate mode ('e' for encryption, 'd' for decryption).
- **Display Result**: Print the result of the encryption or decryption.
- **Repeat or End**: Return to step 1 unless the user chooses to exit.

Code

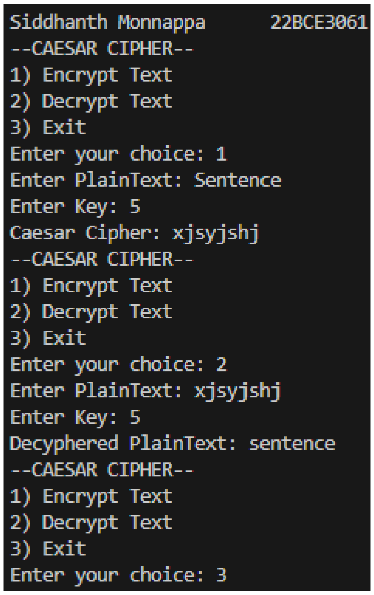
```
def caesar(text, key, chk):
  result=""
  for i in text:
    if i==' ':
      result+=' '
      continue
    if chk == 'e':
      result+=chr((ord(i)+key-97)%26+97)
    elif chk=='d':
      result+=chr((ord(i)-key-97)%26+97)
  return result

print("Siddhanth Monnappa\t22BCE3061")
while(True):
  print("--CAESAR CIPHER--")
print("1) Encrypt Text")
print("2) Decrypt Text")
print("3) Exit")
ch=input("Enter your choice: ")
if ch=='3':
   break
text=input("Enter PlainText: ")
text=text.lower()
key=int(input("Enter Key: "))
if ch=='1':
   print(f"Caesar Cipher: {caesar(text, key, 'e')}")
elif ch=='2':
   print(f"Decyphered PlainText: {caesar(text, key, 'd')}")
```

Code Screenshot



```python
def caesar(text, key, chk):
    result=""
    for i in text:
        if i==' ':
            result+=' '
            continue
        if chk == 'e':
            result+=chr((ord(i)+key-97)%26+97)
        elif chk=='d':
            result+=chr((ord(i)-key-97)%26+97)
    return result

print("Siddhanth Monnappa\t22BCE3061")
while(True):
    print("--CAESAR CIPHER--")
    print("1) Encrypt Text")
    print("2) Decrypt Text")
    print("3) Exit")
    ch=input("Enter your choice: ")
    if ch=='3':
        break
    text=input("Enter PlainText: ")
    text=text.lower()
    key=int(input("Enter Key: "))
    if ch=='1':
        print(f"Caesar Cipher: {caesar(text, key, 'e')}")
    elif ch=='2':
        print(f"Decyphered PlainText: {caesar(text, key, 'd')}")
```

Output



```
Siddhanth Monnappa      22BCE3061
--CAESAR CIPHER--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter your choice: 1
Enter PlainText: Sentence
Enter Key: 5
Caesar Cipher: xjsyjshj
--CAESAR CIPHER--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter your choice: 2
Enter PlainText: xjsyjshj
Enter Key: 5
Decyphered PlainText: sentence
--CAESAR CIPHER--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter your choice: 3
```

Program 2: Playfair Cipher

Pseudocode/Algorithm

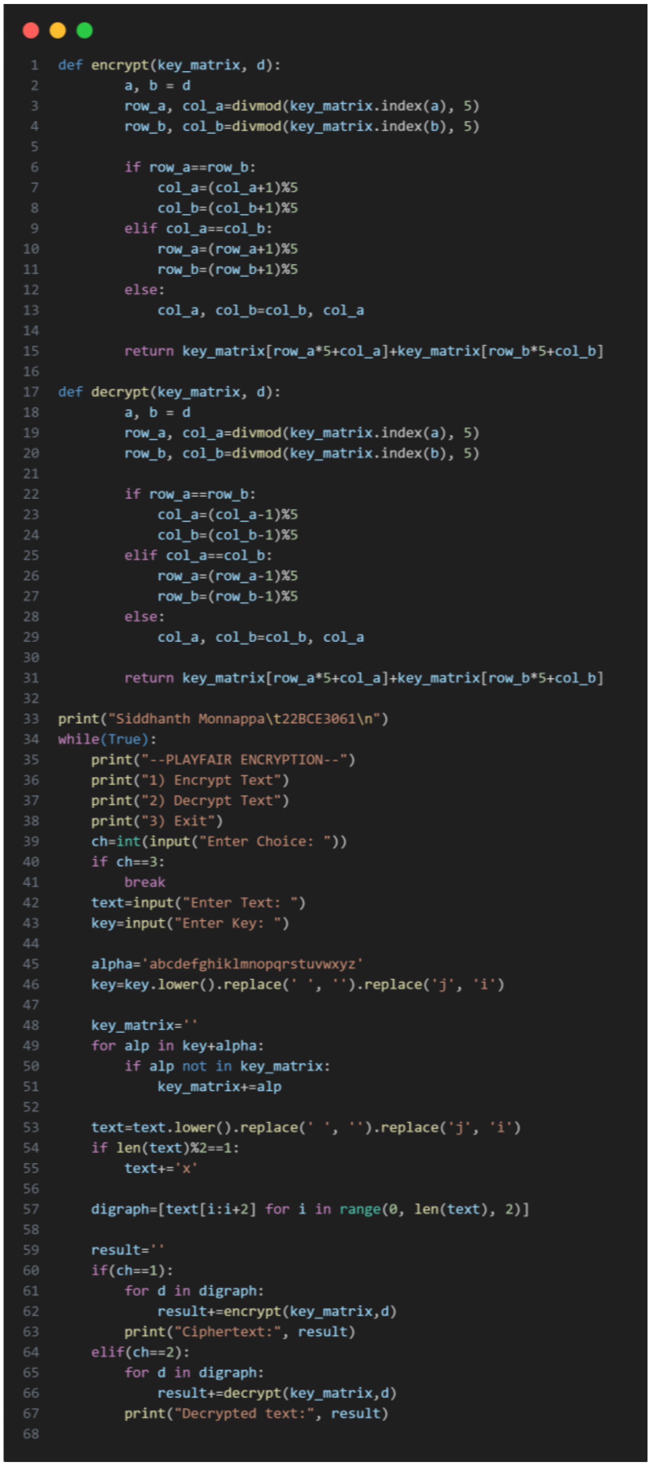
- **Start**: Display the program details and options menu for encryption, decryption, or exit.
- **Input Choice**: Take the user's choice for encryption, decryption, or exit.
- **Exit Check**: If the choice is to exit, terminate the program.
- **Generate Key Matrix**: Take the user-input key, preprocess it, and generate a 5x5 key matrix by combining the key and the alphabet, excluding duplicates and replacing 'j' with 'i'.
- **Preprocess Text**: Preprocess the input text by converting it to lowercase, removing spaces, replacing 'j' with 'i', and appending 'x' if the text length is odd. Split the text into digraphs.
- **Encrypt/Decrypt**: Based on the choice, process each digraph using the encrypt or decrypt function, applying Playfair rules for rows, columns, and rectangles.
- **Display Result**: Print the resulting ciphertext (for encryption) or plaintext (for decryption) and return to step 1 unless the user chooses to exit.

Code

```
def encrypt(key_matrix, d):
    a, b = d
    row_a, col_a=divmod(key_matrix.index(a), 5)
    row_b, col_b=divmod(key_matrix.index(b), 5)


    if row_a==row_b:
      col_a=(col_a+1)%5
      col_b=(col_b+1)%5
    elif col_a==col_b:
      row_a=(row_a+1)%5
      row_b=(row_b+1)%5
    else:
      col_a, col_b=col_b, col_a
    return key_matrix[row_a*5+col_a]+key_matrix[row_b*5+col_b]

def decrypt(key_matrix, d):
    a, b = d
    row_a, col_a=divmod(key_matrix.index(a), 5)
    row_b, col_b=divmod(key_matrix.index(b), 5)

    if row_a==row_b:
      col_a=(col_a-1)%5
      col_b=(col_b-1)%5
    elif col_a==col_b:
      row_a=(row_a-1)%5
      row_b=(row_b-1)%5
    else:
      col_a, col_b=col_b, col_a

    return key_matrix[row_a*5+col_a]+key_matrix[row_b*5+col_b]

print("Siddhanth Monnappa\t22BCE3061\n")
while(True):
  print("--PLAYFAIR ENCRYPTION--")
  print("1) Encrypt Text")
  print("2) Decrypt Text")
  print("3) Exit")
  ch=int(input("Enter Choice: "))
```

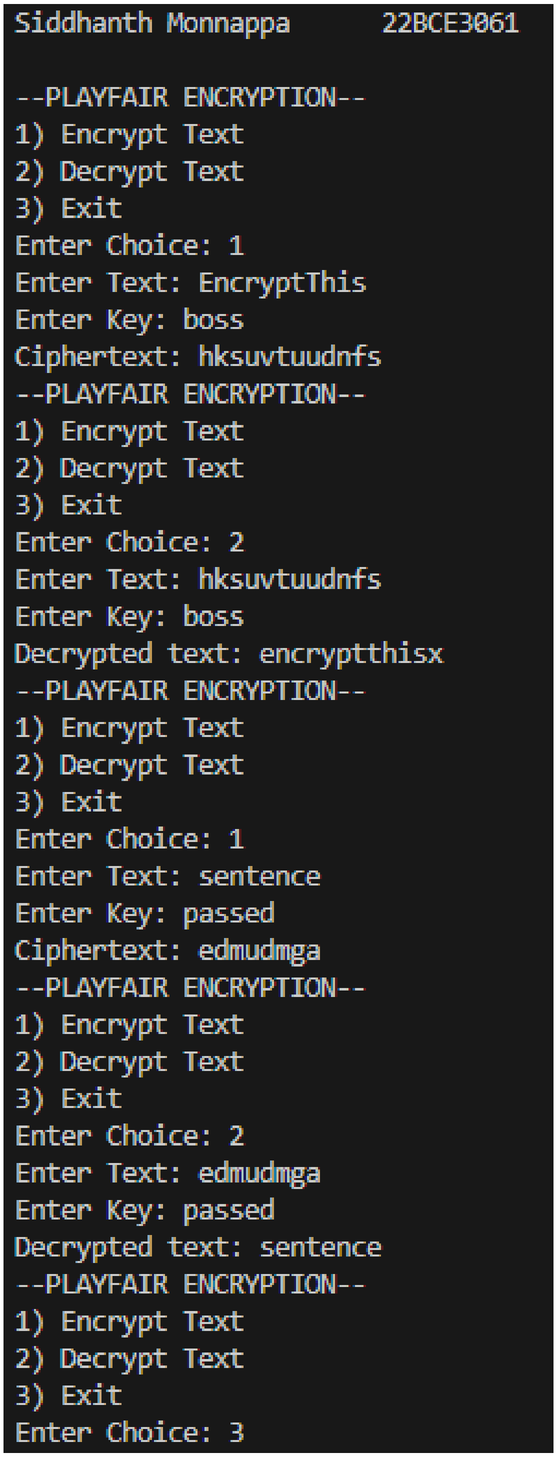
```
if ch==3:
    break
text=input("Enter Text: ")
key=input("Enter Key: ")

alpha='abcdefghiklmnopqrstuvwxyz'
key=key.lower().replace(' ', '').replace('j', 'i')

key_matrix=''
for alp in key+alpha:
    if alp not in key_matrix:
        key_matrix+=alp

text=text.lower().replace(' ', '').replace('j', 'i')
if len(text)%2==1:
    text+='x'

digraph=[text[i:i+2] for i in range(0, len(text), 2)]

result=''
if(ch==1):
    for d in digraph:
        result+=encrypt(key_matrix,d)
    print("Ciphertext:", result)
elif(ch==2):
    for d in digraph:
        result+=decrypt(key_matrix,d)
    print("Decrypted text:", result)
```

Code Screenshot



```
def encrypt(key_matrix, d):
        a, b = d
        row_a, col_a=divmod(key_matrix.index(a), 5)
        row_b, col_b=divmod(key_matrix.index(b), 5)

        if row_a==row_b:
            col_a=(col_a+1)%5
            col_b=(col_b+1)%5
        elif col_a==col_b:
            row_a=(row_a+1)%5
            row_b=(row_b+1)%5
        else:
            col_a, col_b=col_b, col_a

        return key_matrix[row_a*5+col_a]+key_matrix[row_b*5+col_b]

def decrypt(key_matrix, d):
        a, b = d
        row_a, col_a=divmod(key_matrix.index(a), 5)
        row_b, col_b=divmod(key_matrix.index(b), 5)

        if row_a==row_b:
            col_a=(col_a-1)%5
            col_b=(col_b-1)%5
        elif col_a==col_b:
            row_a=(row_a-1)%5
            row_b=(row_b-1)%5
        else:
            col_a, col_b=col_b, col_a

        return key_matrix[row_a*5+col_a]+key_matrix[row_b*5+col_b]

print("Siddhanth Monnappa\t22BCE3061\n")
while(True):
    print("--PLAYFAIR ENCRYPTION--")
    print("1) Encrypt Text")
    print("2) Decrypt Text")
    print("3) Exit")
    ch=int(input("Enter Choice: "))
    if ch==3:
        break
    text=input("Enter Text: ")
    key=input("Enter Key: ")

    alpha='abcdefghiklmnopqrstuvwxyz'
    key=key.lower().replace(' ', '').replace('j', 'i')

    key_matrix=''
    for alp in key+alpha:
        if alp not in key_matrix:
            key_matrix+=alp

    text=text.lower().replace(' ', '').replace('j', 'i')
    if len(text)%2==1:
        text+='x'

    digraph=[text[i:i+2] for i in range(0, len(text), 2)]

    result=''
    if(ch==1):
        for d in digraph:
            result+=encrypt(key_matrix,d)
        print("Ciphertext:", result)
    elif(ch==2):
        for d in digraph:
            result+=decrypt(key_matrix,d)
        print("Decrypted text:", result)
```

Output



```
Siddhanth Monnappa      22BCE3061

--PLAYFAIR ENCRYPTION--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 1
Enter Text: EncryptThis
Enter Key: boss
Ciphertext: hksuvtuudnfs
--PLAYFAIR ENCRYPTION--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 2
Enter Text: hksuvtuudnfs
Enter Key: boss
Decrypted text: encryptthisx
--PLAYFAIR ENCRYPTION--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 1
Enter Text: sentence
Enter Key: passed
Ciphertext: edmudmga
--PLAYFAIR ENCRYPTION--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 2
Enter Text: edmudmga
Enter Key: passed
Decrypted text: sentence
--PLAYFAIR ENCRYPTION--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 3
```

Program 3: Hill Cipher

Pseudocode/Algorithm

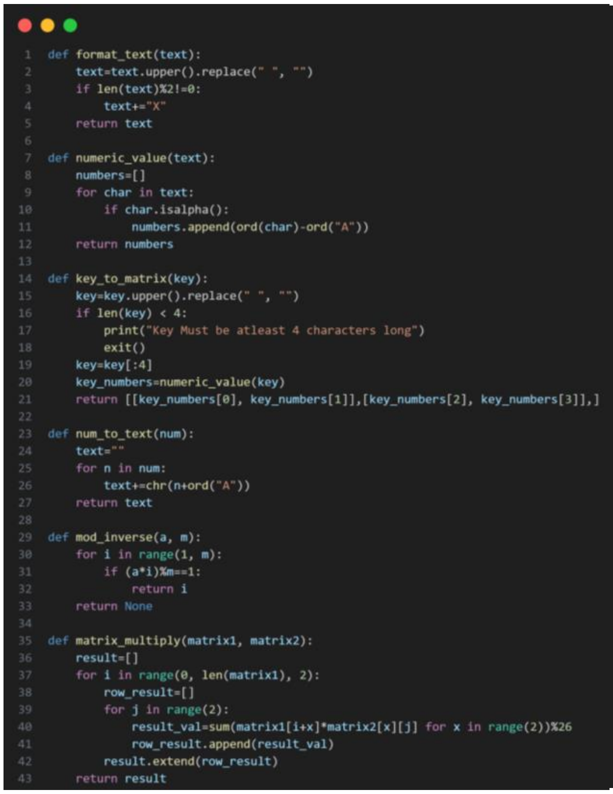
- **Start:** Display the program details and options menu for encryption, decryption, or exit.
- **Input Choice:** Take the user's choice for encryption, decryption, or exit.
- **Exit Check:** If the choice is to exit, terminate the program.
- **Input Key:** Prompt the user for an alphabetic key of at least 4 characters.
- **Input Text:** Prompt the user to encrypt or decrypt.
- **Format Text:** Preprocess the text by converting it to uppercase, removing spaces, and appending "X" if the length is odd.
- **Key Matrix:** Generate a 2x2 key matrix from the first 4 characters of the key, converting letters to numeric values.
- **Encrypt/Decrypt Matrix:** If decrypting, calculate the inverse of the key matrix. If the inverse doesn't exist, terminate with an error.
- **Text to Numbers:** Convert the text into numeric values using their alphabetic positions.
- **Matrix Multiplication:** Multiply the numeric text with the key matrix (for encryption) or its inverse (for decryption), modulo 26.
- **Numbers to Text:** Convert the resulting numbers back to text using their alphabetic positions.
- **Display Result:** Print the resulting ciphertext (for encryption) or plaintext (for decryption) and return to step 1 unless the user chooses to exit.

Code

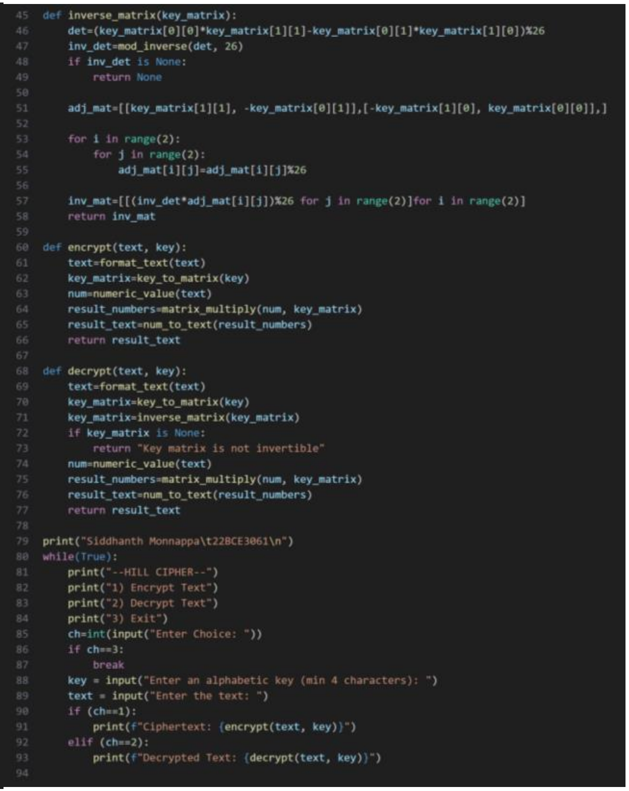
```
def format_text(text):
  text=text.upper().replace(" ", "")
  if len(text)%2!=0:
    text+="X"
  return text


def numeric_value(text):
  numbers=[]
  for char in text:
    if char.isalpha():
          numbers.append(ord(char)-ord("A"))
  return numbers

def key_to_matrix(key):
  key=key.upper().replace(" ", "")
  if len(key) < 4:
    print("Key Must be atleast 4 characters long")
    exit()
  key=key[:4]
  key_numbers=numeric_value(key)
  return [[key_numbers[0], key_numbers[1]],[key_numbers[2], key_numbers[3]],]

def num_to_text(num):
  text=""
  for n in num:
    text+=chr(n+ord("A"))
  return text

def mod_inverse(a, m):
  for i in range(1, m):
    if (a*i)%m==1:
          return i
  return None

def matrix_multiply(matrix1, matrix2):
  result=[]
  for i in range(0, len(matrix1), 2):
    row_result=[]
    for j in range(2):
          result_val=sum(matrix1[i+x]*matrix2[x][j] for x in range(2))%26
          row_result.append(result_val)
    result.extend(row_result)
  return result

def inverse_matrix(key_matrix):
  det=(key_matrix[0][0]*key_matrix[1][1]-key_matrix[0][1]*key_matrix[1][0])%26
  inv_det=mod_inverse(det, 26)
  if inv_det is None:
    return None

  adj_mat=[[key_matrix[1][1], -key_matrix[0][1]],[-key_matrix[1][0], key_matrix[0][0]],]

  for i in range(2):
    for j in range(2):
          adj_mat[i][j]=adj_mat[i][j]%26

  inv_mat=[[(inv_det*adj_mat[i][j])%26 for j in range(2)]for i in range(2)]
  return inv_mat

def encrypt(text, key):
  text=format_text(text)
  key_matrix=key_to_matrix(key)
  num=numeric_value(text)
  result_numbers=matrix_multiply(num, key_matrix)
  result_text=num_to_text(result_numbers)
  return result_text

def decrypt(text, key):
  text=format_text(text)
  key_matrix=key_to_matrix(key)
  key_matrix=inverse_matrix(key_matrix)
  if key_matrix is None:
    return "Key matrix is not invertible"
  num=numeric_value(text)
  result_numbers=matrix_multiply(num, key_matrix)
  result_text=num_to_text(result_numbers)
  return result_text

print("Siddhanth Monnappa\t22BCE3061\n")
while(True):
  print("--HILL CIPHER--")
  print("1) Encrypt Text")
  print("2) Decrypt Text")
  print("3) Exit")
  ch=int(input("Enter Choice: "))
  if ch==3:
    break
  key = input("Enter an alphabetic key (min 4 characters): ")
```

```
text = input("Enter the text: ")

if (ch==1):

  print(f"Ciphertext: {encrypt(text, key)}")

elif (ch==2):

  print(f"Decrypted Text: {decrypt(text, key)}")
```

Code Screenshot

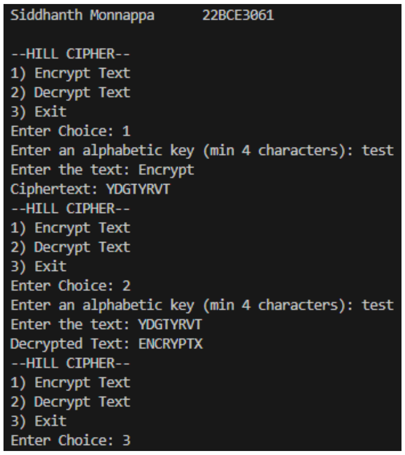


```
def format_text(text):
    text=text.upper().replace(" ", "")
    if len(text)%2!=0:
        text+="X"
    return text

def numeric_value(text):
    numbers=[]
    for char in text:
        if char.isalpha():
            numbers.append(ord(char)-ord("A"))
    return numbers

def key_to_matrix(key):
    key=key.upper().replace(" ", "")
    if len(key) < 4:
        print("Key Must be atleast 4 characters long")
        exit()
    key=key[:4]
    key_numbers=numeric_value(key)
    return [[key_numbers[0], key_numbers[1]],[key_numbers[2], key_numbers[3]],]

def num_to_text(num):
    text=""
    for n in num:
        text+=chr(n+ord("A"))
    return text

def mod_inverse(a, m):
    for i in range(1, m):
        if (a*i)%m==1:
            return i
    return None

def matrix_multiply(matrix1, matrix2):
    result=[]
    for i in range(0, len(matrix1), 2):
        row_result=[]
        for j in range(2):
            result_val=sum(matrix1[i+x]*matrix2[x][j] for x in range(2))%26
            row_result.append(result_val)
        result.extend(row_result)
    return result
```

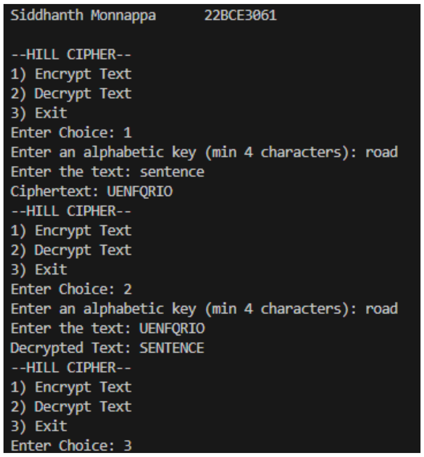


```
def inverse_matrix(key_matrix):
    det=(key_matrix[0][0]*key_matrix[1][1]-key_matrix[0][1]*key_matrix[1][0])%26
    inv_det=mod_inverse(det, 26)
    if inv_det is None:
        return None

    adj_mat=[[key_matrix[1][1], -key_matrix[0][1]],[-key_matrix[1][0], key_matrix[0][0]],]

    for i in range(2):
        for j in range(2):
            adj_mat[i][j]=adj_mat[i][j]%26

    inv_mat=[[(inv_det*adj_mat[i][j])%26 for j in range(2)]for i in range(2)]
    return inv_mat

def encrypt(text, key):
    text=format_text(text)
    key_matrix=key_to_matrix(key)
    num=numeric_value(text)
    result_numbers=matrix_multiply(num, key_matrix)
    result_text=num_to_text(result_numbers)
    return result_text

def decrypt(text, key):
    text=format_text(text)
    key_matrix=key_to_matrix(key)
    key_matrix=inverse_matrix(key_matrix)
    if key_matrix is None:
        return "Key matrix is not invertible"
    num=numeric_value(text)
    result_numbers=matrix_multiply(num, key_matrix)
    result_text=num_to_text(result_numbers)
    return result_text

print("Siddhanth Monnappa\t22BCE3061\n")
while(True):
    print("--HILL CIPHER--")
    print("1) Encrypt Text")
    print("2) Decrypt Text")
    print("3) Exit")
    ch=int(input("Enter Choice: "))
    if ch==3:
        break
    key = input("Enter an alphabetic key (min 4 characters): ")
    text = input("Enter the text: ")
    if (ch==1):
        print(f"Ciphertext: {encrypt(text, key)}")
    elif (ch==2):
        print(f"Decrypted Text: {decrypt(text, key)}")
```

Output



```
Siddhanth Monnappa      22BCE3061

--HILL CIPHER--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 1
Enter an alphabetic key (min 4 characters): test
Enter the text: Encrypt
Ciphertext: YDGTYRVT
--HILL CIPHER--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 2
Enter an alphabetic key (min 4 characters): test
Enter the text: YDGTYRVT
Decrypted Text: ENCRYPTX
--HILL CIPHER--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 3
```



```
Siddhanth Monnappa      22BCE3061

--HILL CIPHER--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 1
Enter an alphabetic key (min 4 characters): road
Enter the text: sentence
Ciphertext: UENFQRIO
--HILL CIPHER--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 2
Enter an alphabetic key (min 4 characters): road
Enter the text: UENFQRIO
Decrypted Text: SENTENCE
--HILL CIPHER--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 3
```

Program 4: Vigenère Cipher

Pseudocode/Algorithm

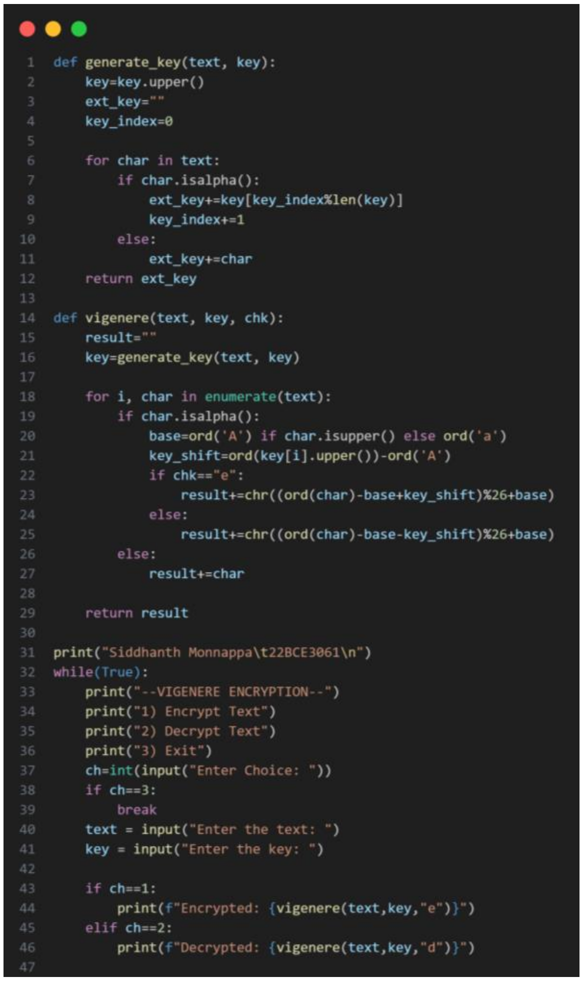
- **Start**: Display the program details and options menu for encryption, decryption, or exit.
- **Input Choice**: Take the user's choice for encryption, decryption, or exit.
- **Exit Check**: If the choice is to exit, terminate the program.
- **Input Text and Key**: Prompt the user to input the text and the key for encryption or decryption.
- **Generate Key**: Extend the key to match the length of the text, keeping non-alphabetic characters in place.
- **Iterate Over Text**: Loop through each character in the text.
- **Check Alphabetic Characters**: If the character is alphabetic, calculate its new position based on the key and mode (encrypt or decrypt).
- **Handle Non-Alphabetic Characters**: If the character is not alphabetic, retain it in the result.
- **Build Result**: Combine the processed characters to form the encrypted or decrypted text.
- **Display Result**: Print the resulting ciphertext or plaintext and return to step 1 unless the user chooses to exit.

Code

```
def generate_key(text, key):
    key=key.upper()
    ext_key=""
    key_index=0

    for char in text:
        if char.isalpha():
            ext_key+=key[key_index%len(key)]
            key_index+=1
        else:
            ext_key+=char
    return ext_key


def vigenere(text, key, chk):
    result=""
    key=generate_key(text, key)

    for i, char in enumerate(text):
        if char.isalpha():
            base=ord('A') if char.isupper() else ord('a')
            key_shift=ord(key[i].upper())-ord('A')
            if chk=="e":
                result+=chr((ord(char)-base+key_shift)%26+base)
            else:
                result+=chr((ord(char)-base-key_shift)%26+base)
        else:
            result+=char

    return result


print("Siddhanth Monnappa\t22BCE3061\n")
while(True):
    print("--VIGENERE ENCRYPTION--")
    print("1) Encrypt Text")
    print("2) Decrypt Text")
    print("3) Exit")
    ch=int(input("Enter Choice: "))
```

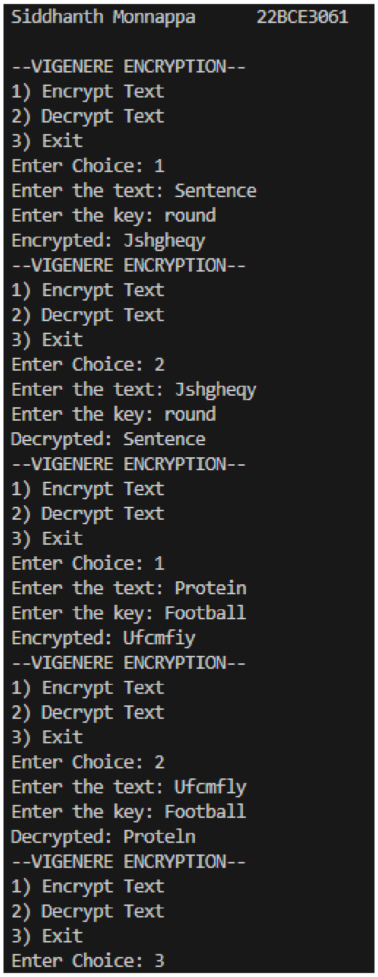
```
if ch==3:
   break
text = input("Enter the text: ")
key = input("Enter the key: ")


if ch==1:
   print(f"Encrypted: {vigenere(text,key,"e")}")
elif ch==2:
   print(f"Decrypted: {vigenere(text,key,"d")}")
```

Code Screenshot



```
def generate_key(text, key):
    key=key.upper()
    ext_key=""
    key_index=0

    for char in text:
        if char.isalpha():
            ext_key+=key[key_index%len(key)]
            key_index+=1
        else:
            ext_key+=char
    return ext_key

def vigenere(text, key, chk):
    result=""
    key=generate_key(text, key)

    for i, char in enumerate(text):
        if char.isalpha():
            base=ord('A') if char.isupper() else ord('a')
            key_shift=ord(key[i].upper())-ord('A')
            if chk=="e":
                result+=chr((ord(char)-base+key_shift)%26+base)
            else:
                result+=chr((ord(char)-base-key_shift)%26+base)
        else:
            result+=char

    return result

print("Siddhanth Monnappa\t22BCE3061\n")
while(True):
    print("--VIGENERE ENCRYPTION--")
    print("1) Encrypt Text")
    print("2) Decrypt Text")
    print("3) Exit")
    ch=int(input("Enter Choice: "))
    if ch==3:
        break
    text = input("Enter the text: ")
    key = input("Enter the key: ")

    if ch==1:
        print(f"Encrypted: {vigenere(text,key,"e")}")
    elif ch==2:
        print(f"Decrypted: {vigenere(text,key,"d")}")
```

Output



```
Siddhanth Monnappa      22BCE3061

--VIGENERE ENCRYPTION--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 1
Enter the text: Sentence
Enter the key: round
Encrypted: Jshgheqy
--VIGENERE ENCRYPTION--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 2
Enter the text: Jshgheqy
Enter the key: round
Decrypted: Sentence
--VIGENERE ENCRYPTION--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 1
Enter the text: Protein
Enter the key: Football
Encrypted: Ufcmfiy
--VIGENERE ENCRYPTION--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 2
Enter the text: Ufcmfly
Enter the key: Football
Decrypted: Proteln
--VIGENERE ENCRYPTION--
1) Encrypt Text
2) Decrypt Text
3) Exit
Enter Choice: 3
```