CNS LAB 5

SAHIL BONDRE: U18CO021

Write a menu-driven program with appropriate functions to implement the affine cipher i.e. E(x) = (a x + b) mod 26. Let the values of a and b be entered by the user. Your program must check for the feasibility of these values before encrypting the plaintext. The program must also output the decrypted values. Let the plaintext be input as a character array of defined size.

```
def recursive_read(allowed_input, message=""):
    # Recursively reads user input until input is not in allowd_input
    while True:
        user_input = input(message)
        if user_input in allowed_input:
            return user_input



def recursive_read_int(message=""):
    # Recursively reads user input until input is not in allowd_input
    while True:
        user_input = input(message)
        try:
            value = int(user_input)
            return value
        except:
            pass
```

```
allowed_a = [1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25]
m = 26



def multiplicative_inverse(a):
    for i in range(1, m):
        remainder = ((i * m) + 1) % a
        if remainder == 0:
            return ((i * m) + 1) / a

    return 0



def perform_encryption():
    message = input("Enter message: ").upper()
    a = recursive_read_int("Enter a: ")
    if a not in allowed_a:
        print("Error: a should be coprimes with m (26)")
        exit()

    b = recursive_read_int("Enter b: ")

    if b >= m:
        print("b should be between 0 to m - 1 (25). Taking modulo 26")
```

```
    b = b % m

    result = ""

    for c in message:
        if c.isalpha():
            x = ord(c) - ord("A")
            e = (a * x + b) % m
            result += chr(ord("A") + e)
        else:
            result += c

    print(f"Final string:\n{result}")



def perform_decryption():
    message = input("Enter message: ").upper()
    a = recursive_read_int("Enter a: ")
    if a not in allowed_a:
        print("Error: a should be coprimes with m (26)")
        exit()

    b = recursive_read_int("Enter b: ")

    if b >= m:
        print("b should be between 0 to m - 1 (25). Taking modulo 26")
```

```
    b = b % m

    a_inv = multiplicative_inverse(a)
    result = ""

    for c in message:
        if c.isalpha():
            x = ord(c) - ord("A")
            d = (a_inv * (x - b)) % m
            result += chr(ord("A") + int(d))
        else:
            result += c

    print(f"Final string:\n{result}")



is_encrypt = recursive_read(
    ["e", "d"], "Enter 'e' for encryption or 'd' for decryption: ") == "e"



if is_encrypt:
    perform_encryption()
else:
    perform_decryption()
```

```
PS F:\code\github.com\godcrampy\college-notes\cns\lab-05> python.exe .\affine.py
Enter 'e' for encryption or 'd' for decryption: e
Enter message: ITS COOL
Enter a: 5
Enter b: 8
Final string:
WZU SAAL
PS F:\code\github.com\godcrampy\college-notes\cns\lab-05> python.exe .\affine.py
Enter 'e' for encryption or 'd' for decryption: d
Enter message: WZU SAAL
Enter a: 5
Enter b: 8
Final string:
ITS COOL
PS F:\code\github.com\godcrampy\college-notes\cns\lab-05>
```