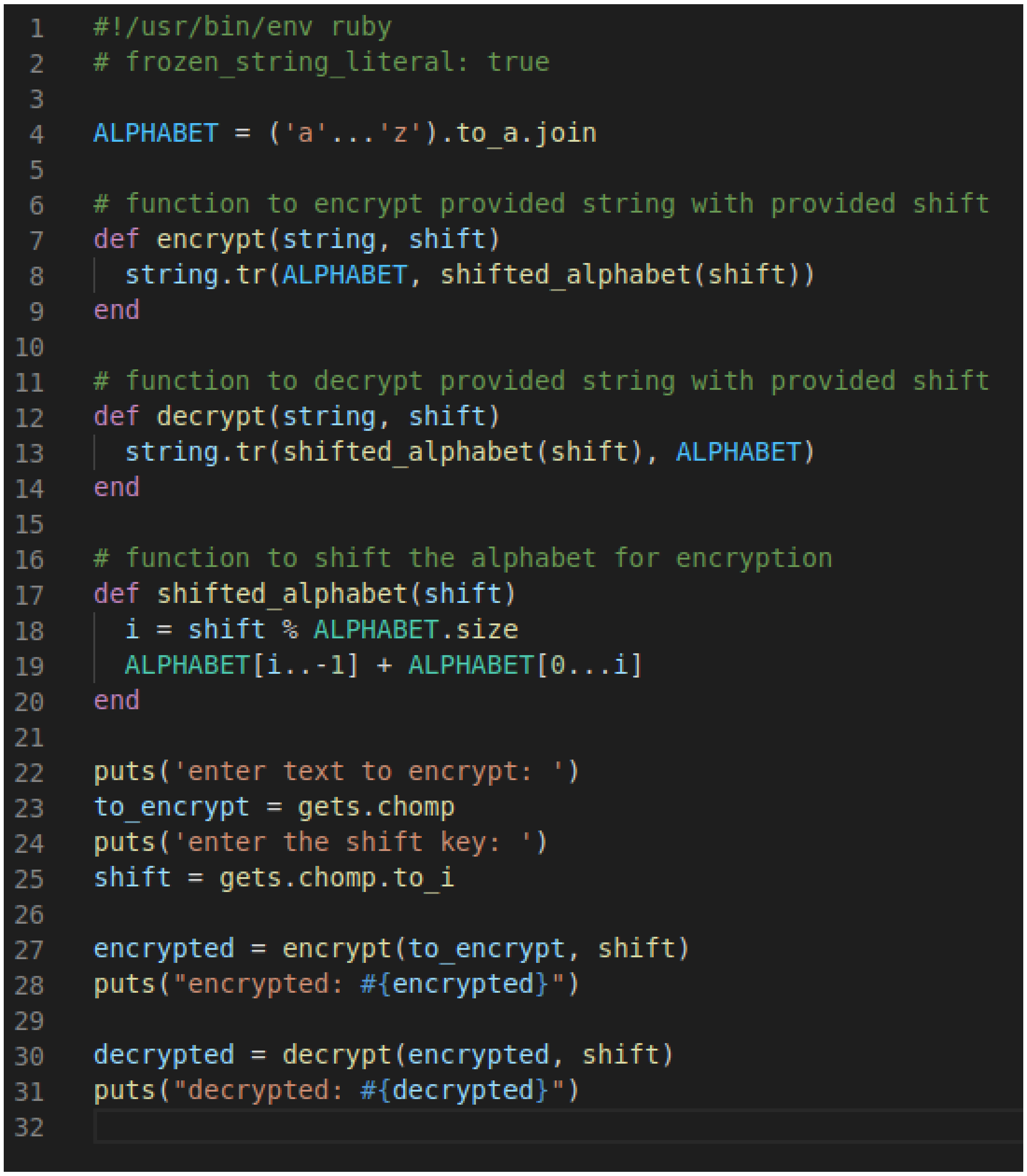
TASK 2

Caesar cipher

Both programs will be from information security class from last semester.

First program is the Caesar cipher (link to the program made in previous semester: https://github.com/SauleStan/IS/tree/master/caeserCipher)

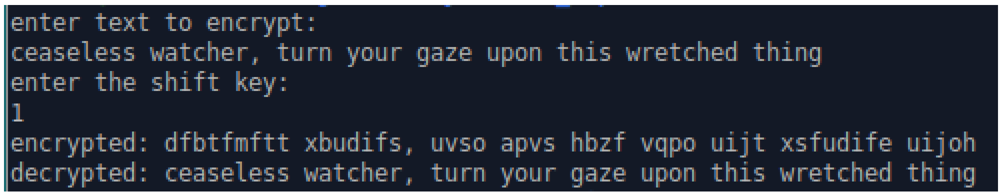


```ruby
#!/usr/bin/env ruby
# frozen_string_literal: true

ALPHABET = ('a'...'z').to_a.join

# function to encrypt provided string with provided shift
def encrypt(string, shift)
  string.tr(ALPHABET, shifted_alphabet(shift))
end

# function to decrypt provided string with provided shift
def decrypt(string, shift)
  string.tr(shifted_alphabet(shift), ALPHABET)
end

# function to shift the alphabet for encryption
def shifted_alphabet(shift)
  i = shift % ALPHABET.size
  ALPHABET[i..-1] + ALPHABET[0...i]
end

puts('enter text to encrypt: ')
to_encrypt = gets.chomp
puts('enter the shift key: ')
shift = gets.chomp.to_i

encrypted = encrypt(to_encrypt, shift)
puts("encrypted: #{encrypted}")

decrypted = decrypt(encrypted, shift)
puts("decrypted: #{decrypted}")

```

Figure 1 - entire program

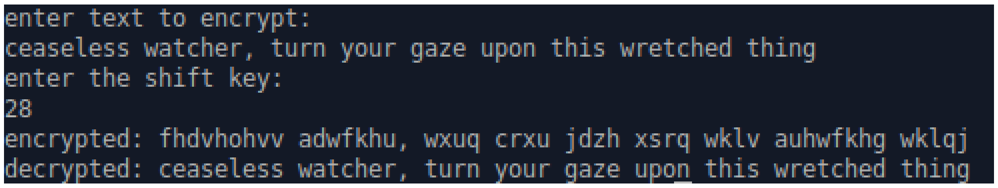
The code has three functions. One to encrypt the given string with given shift key, another to decrypt the encrypted string, and third one to shift the encryption alphabet. Both encrypt and decrypt functions use the tr() function on given string to substitute letters in the string by the shifted alphabet. Shifted alphabet is provided by the shift_alphabet() function that accepts a shift value and rearranges the alphabet to start from the index where shift value is to the end of alphabet and adding to the end the values from beginning of the alphabet to the shift value, returning shifted alphabet for encryption.

When run, the script should ask the user to input the text for encryption and provide shift key. The output shows the result of encryption and decryption of encrypted text.



```
enter text to encrypt:
ceaseless watcher, turn your gaze upon this wretched thing
enter the shift key:
1
encrypted: dfbtfmftt xbudifs, uvso apvs hbzf vqpo uijt xsfudife uijoh
decrypted: ceaseless watcher, turn your gaze upon this wretched thing
```

Figure 2 - output 1

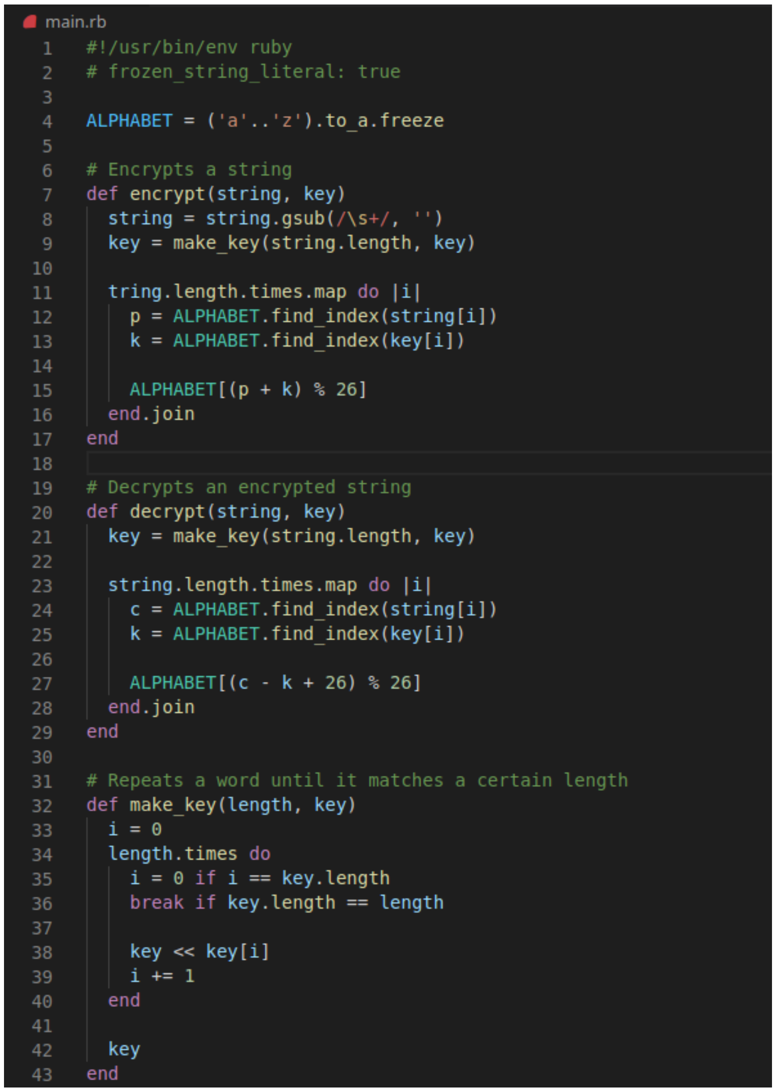


```
enter text to encrypt:
ceaseless watcher, turn your gaze upon this wretched thing
enter the shift key:
28
encrypted: fhdvhohvv adwfkhu, wxuq crxu jdzh xsrq wklv auhwfkhg wklqj
decrypted: ceaseless watcher, turn your gaze upon this wretched thing
```

Figure 3 - output 2

Vigenere cipher

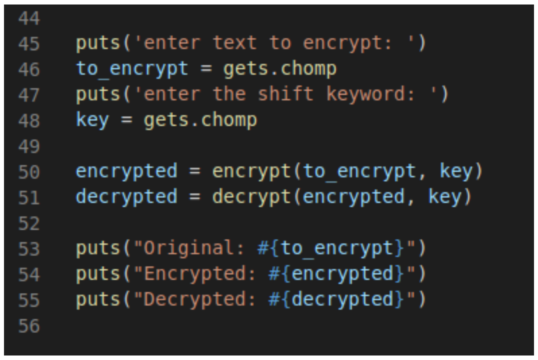
Second program is the Vigenere cipher (link to the program made in previous semester: https://github.com/SauleStan/IS/tree/master/Vigenere)



main.rb

```ruby
#!/usr/bin/env ruby
# frozen_string_literal: true

ALPHABET = ('a'..'z').to_a.freeze

# Encrypts a string
def encrypt(string, key)
  string = string.gsub(/\s+/, '')
  key = make_key(string.length, key)

  tring.length.times.map do |i|
    p = ALPHABET.find_index(string[i])
    k = ALPHABET.find_index(key[i])

    ALPHABET[(p + k) % 26]
  end.join
end

# Decrypts an encrypted string
def decrypt(string, key)
  key = make_key(string.length, key)

  string.length.times.map do |i|
    c = ALPHABET.find_index(string[i])
    k = ALPHABET.find_index(key[i])

    ALPHABET[(c - k + 26) % 26]
  end.join
end

# Repeats a word until it matches a certain length
def make_key(length, key)
  i = 0
  length.times do
    i = 0 if i == key.length
    break if key.length == length

    key << key[i]
    i += 1
  end

  key
end
```

Figure 4 - vigenere functions

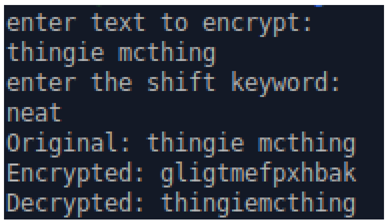


```ruby

puts('enter text to encrypt: ')
to_encrypt = gets.chomp
puts('enter the shift keyword: ')
key = gets.chomp

encrypted = encrypt(to_encrypt, key)
decrypted = decrypt(encrypted, key)

puts("Original: #{to_encrypt}")
puts("Encrypted: #{encrypted}")
puts("Decrypted: #{decrypted}")

```

Figure 5 - output part of the program

There are three functions in this program, one to encrypt, one to decrypt and one to make the key for encryption. The make_key function takes in the length of the string that needs to be encrypted and the provided encryption word, then returns the keyword’s letters repeated the same amount as the string for encryption. Encryption function takes the string to be encrypted, uses the make_key function to copy the keyword as many times as needed for the string’s length and then for each letter adds them with mod 26 (alphabet’s length). Decrypt function does mostly the same thing as encryption, only the calculation part for each letter’s shift differs.

When run, the script should ask the user to input the text for encryption and provide encryption keyword. The output shows the result of encryption and decryption of encrypted text.



```
enter text to encrypt:
thingie mcthing
enter the shift keyword:
neat
Original: thingie mcthing
Encrypted: gligtmefpxhbak
Decrypted: thingiemcthing
```

Figure 6 – output