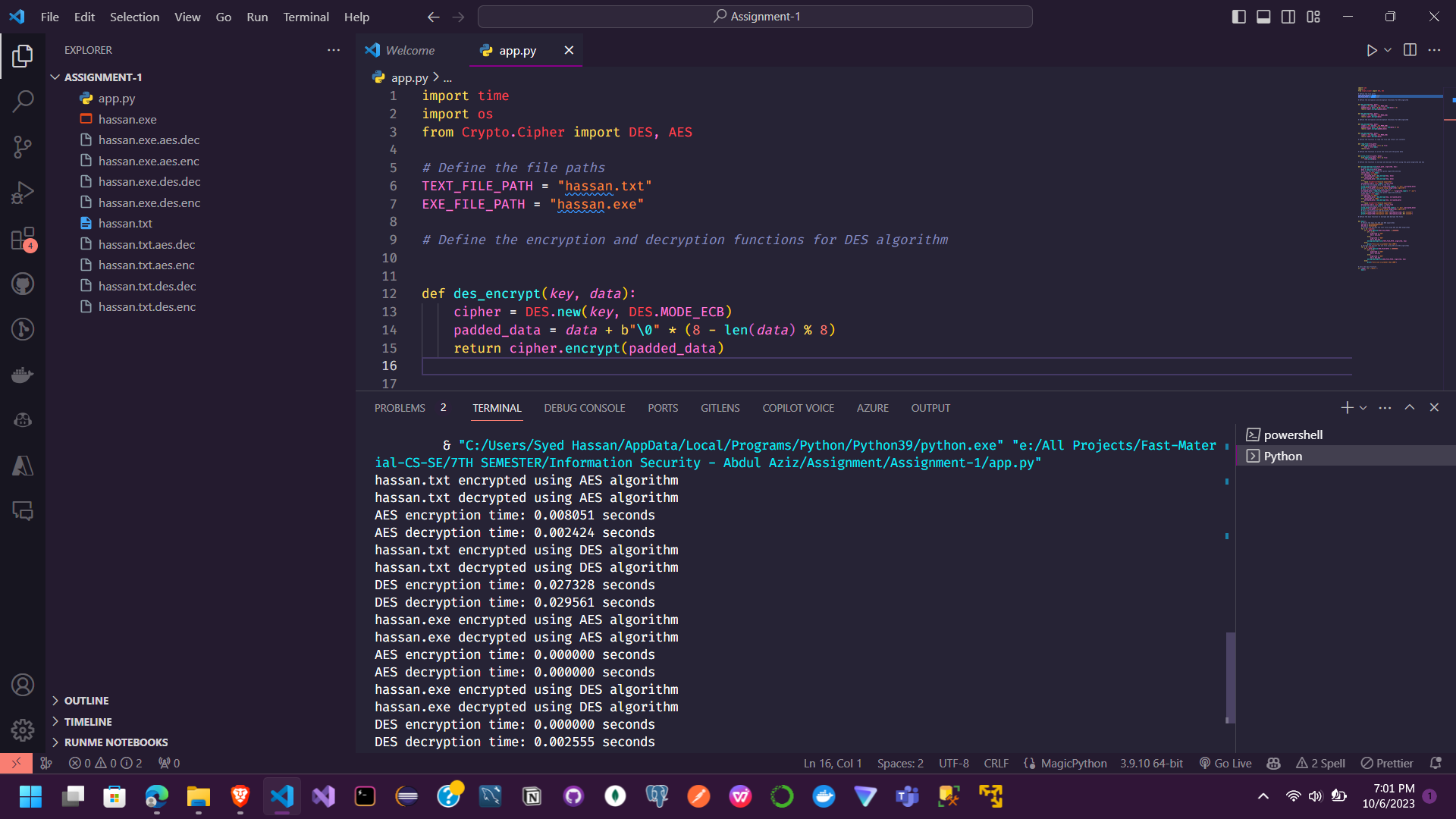
**IS - ASSIGNMENT - 1**

**HASSAN ALI**

**K20-1052**

**BSE-7B**

**Encryption and Decryption of .txt and .exe file using DES and AES**



import time

import os

from Crypto.Cipher import DES, AES

*# Define the file paths*

TEXT\_FILE\_PATH = "hassan.txt"

EXE\_FILE\_PATH = "hassan.exe"

*# Define the encryption and decryption functions for DES algorithm*

def des\_encrypt(*key*, *data*):

    cipher = DES.new(*key*, DES.MODE\_ECB)

    padded\_data = *data* + b"\0" \* (8 - len(*data*) % 8)

    return cipher.encrypt(padded\_data)

def des\_decrypt(*key*, *data*):

    cipher = DES.new(*key*, DES.MODE\_ECB)

    return cipher.decrypt(*data*)

*# Define the encryption and decryption functions for AES algorithm*

def aes\_encrypt(*key*, *data*):

    cipher = AES.new(*key*, AES.MODE\_ECB)

    padded\_data = *data* + b"\0" \* (16 - len(*data*) % 16)

    return cipher.encrypt(padded\_data)

def aes\_decrypt(*key*, *data*):

    cipher = AES.new(*key*, AES.MODE\_ECB)

    return cipher.decrypt(*data*)

*# Define the function to read the file and return its contents*

def read\_file(*file\_path*):

    with open(*file\_path*, "rb") as file:

        data = file.read()

    return data

*# Define the function to write the file with the given data*

def write\_file(*file\_path*, *data*):

    with open(*file\_path*, "wb") as file:

        file.write(*data*)

*# Define the function to encrypt and decrypt the file using the given algorithm and key*

def encrypt\_decrypt\_file(*file\_path*, *algorithm*, *key*):

*# Read the file contents*

    data = read\_file(*file\_path*)

*# Encrypt the data using the given algorithm and key*

    start\_time = time.time()

    if *algorithm* == "DES":

        encrypted\_data = des\_encrypt(*key*, data)

    elif *algorithm* == "AES":

        encrypted\_data = aes\_encrypt(*key*, data)

    else:

        raise ValueError("Invalid algorithm")

    encryption\_time = time.time() - start\_time

*# Write the encrypted data to the file*

    write\_file(*file\_path* + "." + *algorithm*.lower() + ".enc", encrypted\_data)

    print(f"{*file\_path*} encrypted using {*algorithm*} algorithm")

*# Read the encrypted data from the file*

    encrypted\_data = read\_file(*file\_path* + "." + *algorithm*.lower() + ".enc")

*# Decrypt the data using the given algorithm and key*

    start\_time = time.time()

    if *algorithm* == "DES":

        decrypted\_data = des\_decrypt(*key*, encrypted\_data)

    elif *algorithm* == "AES":

        decrypted\_data = aes\_decrypt(*key*, encrypted\_data)

    else:

        raise ValueError("Invalid algorithm")

    decryption\_time = time.time() - start\_time

*# Write the decrypted data to the file*

    write\_file(*file\_path* + "." + *algorithm*.lower() + ".dec", decrypted\_data)

    print(f"{*file\_path*} decrypted using {*algorithm*} algorithm")

*# Print the encryption and decryption times*

    print(f"{*algorithm*} encryption time: {encryption\_time:.6f} seconds")

    print(f"{*algorithm*} decryption time: {decryption\_time:.6f} seconds")

*# Define the main function to encrypt and decrypt the files*

def main():

*# Define the keys for AES and DES algorithms*

    aes\_key = b"1234567890123456"

    des\_key = b"abcdefgh"

*# Encrypt and decrypt the text file using AES and DES algorithms*

    for i in range(2):

        if os.path.getsize(TEXT\_FILE\_PATH) < 10000000:

            if i == 0:

                algorithm = "AES"

                key = aes\_key

            else:

                algorithm = "DES"

                key = des\_key

            encrypt\_decrypt\_file(TEXT\_FILE\_PATH, algorithm, key)

        else:

            print("File size is greater than 10MB")

*# Encrypt and decrypt the exe file using AES and DES algorithms*

    for i in range(2):

        if os.path.getsize(EXE\_FILE\_PATH) < 10000000:

            if i == 0:

                algorithm = "AES"

                key = aes\_key

            else:

                algorithm = "DES"

                key = des\_key

            encrypt\_decrypt\_file(EXE\_FILE\_PATH, algorithm, key)

        else:

            print("File size is greater than 10MB")

*# Call the main function*

if \_\_name\_\_ == "\_\_main\_\_":

    main()