**Name: Tayyab Zain**

**Reg No: 4293-FBAS/BSCS/F20**

**Assignment 1:**

**Code:**  
#include <iostream>

#include <string>

using namespace std;

string CipheredText = "";

string Decryption(int decrpt\_key) {

string deciphereText = "";

for (int i = 0; i < CipheredText.length(); i++) {

int letter = int(CipheredText[i]) - decrpt\_key;

char new\_letter = char(letter);

deciphereText += new\_letter;

}

return deciphereText;

}

static int encrypt\_key = 0;

int main() {

while (true) {

string plaintext;

cout << "Enter the plain text: ";

cin.ignore();

getline(cin, plaintext);

if (plaintext.length() == 0) {

cout << "Error: Empty string was entered" << endl;

}

else {

cout << "Enter Key: ";

cin >> encrypt\_key;

cin.ignore(); // Ignore the remaining newline character

for (int i = 0; i < plaintext.length(); i++) {

int letter = int(plaintext[i]) + encrypt\_key;

char new\_letter = char(letter);

CipheredText += new\_letter;

}

}

cout << "Plain Text: " << plaintext << endl;

cout << "Ciphered Text: " << CipheredText << endl;

cout << "Decryption: " << endl;

int decrpt\_key = 0;

cout << "Enter Decryption Key: ";

cin >> decrpt\_key;

if (decrpt\_key == encrypt\_key) {

cout << "Correct Key" << endl;

cout << "Deciphered Text" << endl;

cout << "Ciphered Text: " << CipheredText << endl;

cout << "Deciphered Text: " << Decryption(decrpt\_key) << endl;

}

else {

cout << "Wrong Key" << endl;

cout << "Deciphered Text" << endl;

cout << "Ciphered Text: " << CipheredText << endl;

cout << "Deciphered Text: " << Decryption(decrpt\_key) << endl;

}

// Clear CipheredText for the next iteration

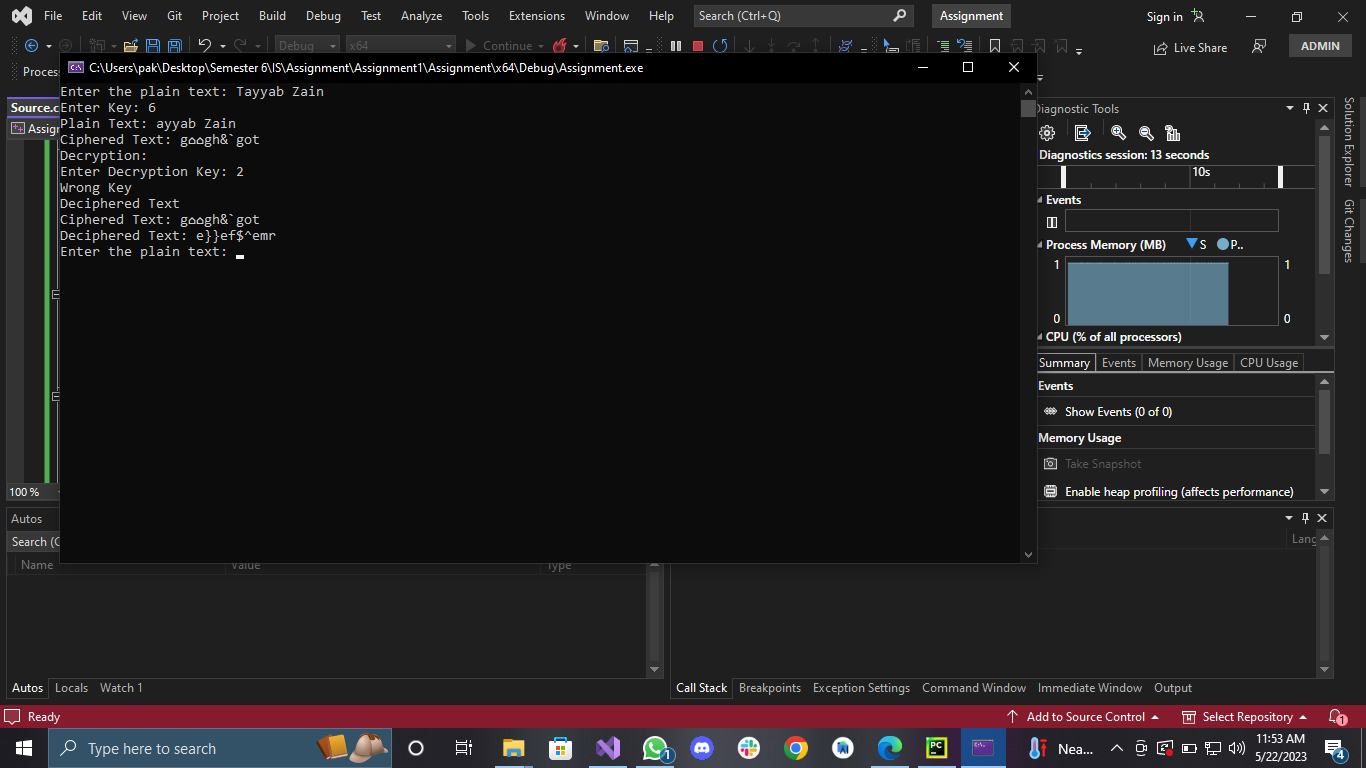
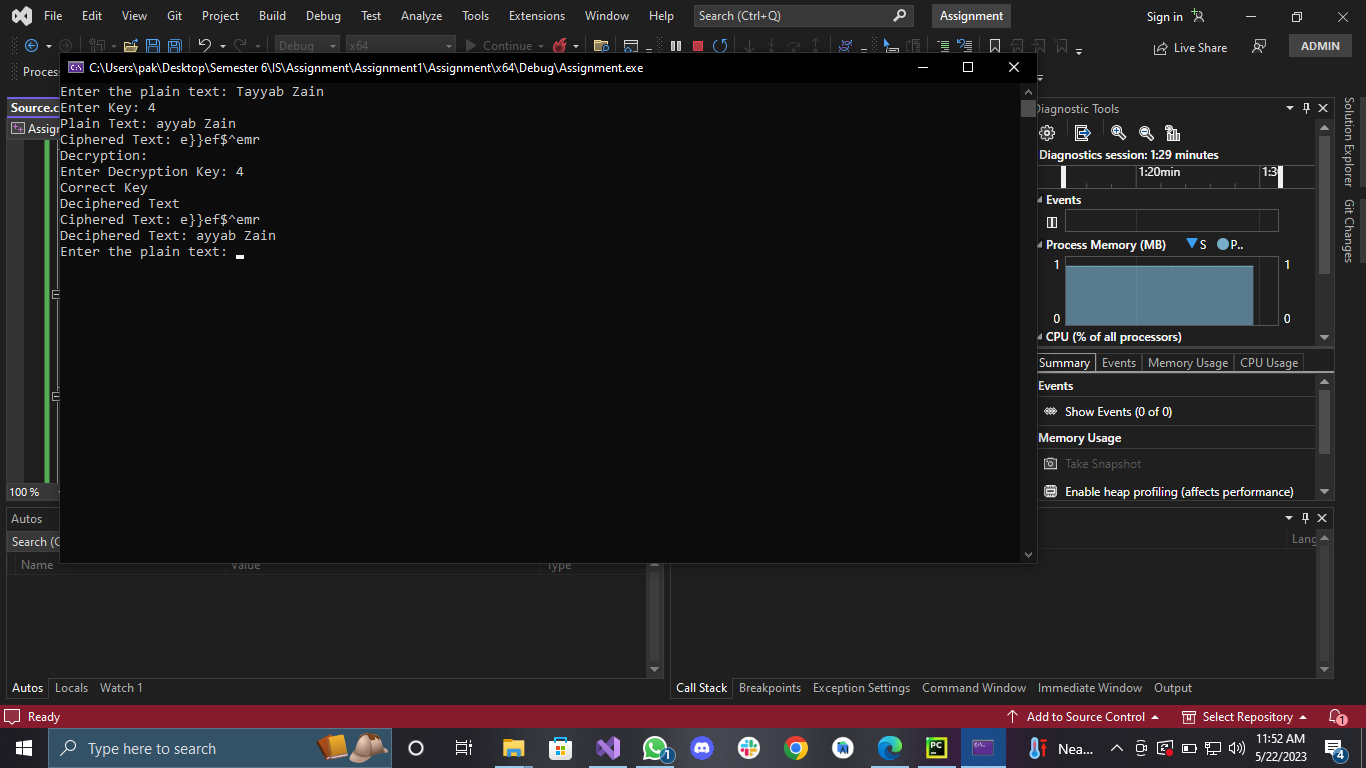
CipheredText = "";

}

return 0;

}

Output:



**Assignment 2:  
Code:**

#include <iostream>

#include <string>

#include <random>

#include <ctime>

using namespace std;

string generateRandomKey(int length) {

string key = "";

const string alphabet = "abcdefghijklmnopqrstuvwxyz";

mt19937 rng(time(0));

uniform\_int\_distribution<int> distribution(0, alphabet.size() - 1);

for (int i = 0; i < length; i++) {

int index = distribution(rng);

key += alphabet[index];

}

return key;

}

string encryptMessage(const string& message, const string& key) {

string encryptedMessage = message;

for (int i = 0; i < encryptedMessage.length(); i++) {

char c = encryptedMessage[i];

if (isalpha(c)) {

char base = islower(c) ? 'a' : 'A';

int shift = key[i % key.length()] - 'a';

encryptedMessage[i] = ((c - base + shift) % 26) + base;

}

}

return encryptedMessage;

}

string decryptMessage(const string& encryptedMessage, const string& key) {

string decryptedMessage = encryptedMessage;

for (int i = 0; i < decryptedMessage.length(); i++) {

char c = decryptedMessage[i];

if (isalpha(c)) {

char base = islower(c) ? 'a' : 'A';

int shift = key[i % key.length()] - 'a';

decryptedMessage[i] = ((c - base - shift + 26) % 26) + base;

}

}

return decryptedMessage;

}

int main() {

string message;

cout << "Enter a message: ";

getline(cin, message);

string key = generateRandomKey(message.length());

cout << "Randomly generated key: " << key << endl;

string encryptedMessage = encryptMessage(message, key);

cout << "Encrypted message: " << encryptedMessage << endl;

string decryptedMessage = decryptMessage(encryptedMessage, key);

cout << "Decrypted message: " << decryptedMessage << endl;

return 0;

}

**Output:**