```
/*
 Name: David Giacobbi
 Class: CPSC 122, Section 01
 Date Submitted: February 3, 2022
 Assignment: Project 4

 Description: This project takes encrypts and decrypts
              a file using the substitution cipher.
*/

#include <iostream>
#include <fstream>
using namespace std;

void runMode(fstream&, fstream&, fstream&, int);
int keyGen();
char encrypt(char, int);
char decrypt(char, int);
void fileOpen(fstream&, string, char);

int main(int argc, char* argv[])
{
    fstream keyFile, inFile, outFile;

    int userMode;
    userMode = atoi(argv[1]);

    if (userMode == 0)
    {
        fileOpen(keyFile, argv[2], 'w');
        int key = keyGen();
        keyFile << key;

    }
    else
    {
        fileOpen(keyFile, argv[2], 'r');
        fileOpen(inFile, argv[3], 'r');
        fileOpen(outFile, argv[4], 'w');

        runMode(keyFile, inFile, outFile, userMode);
    }

    keyFile.close();
    inFile.close();
    outFile.close();
}

/*
Description: Runs the requested encryption or decryption based
             on mode choice in command line.
```

```
Input: int, number that represents key generation, encryption,
       or decryption
Output: N/A
*/
void runMode(fstream& keyFile, fstream& inFile, fstream& outFile, int mode)
{
    // Variable declaration
    char ch;
    int key;

    // Encryption sequence
    if (mode == 1)
    {
        // Store key from keyFile
        keyFile >> key;

        // While loop encrypts characters one by one using function
        // only if they are alphabetical
        while (inFile.peek() != EOF)
        {
            ch = inFile.get();
            if (isalpha(ch))
            {
                // Changes character to uppercase ASCII values before encrypt
                ch = toupper(ch);
                ch = encrypt(ch, key);
            }
            outFile.put(ch);
        }
    }
    // Decryption sequence (same as encryption but with decrypt function)
    else if (mode == 2)
    {
        keyFile >> key;

        while (inFile.peek() != EOF)
        {
            ch = inFile.get();
            if (isalpha(ch))
                ch = decrypt(ch, key);
            outFile.put(ch);
        }
    }
    // Error checking for mode inputs not 0-2
    else
    {
        cout << "Please enter a valid mode of operation choice." << endl;
        exit(EXIT_FAILURE);
    }
}
```

```
/*
Description: Randomly generates an integer in the range: [1..25]
Input: none
Output: returns a randomly generated integer in the range [1..25]
*/
int keyGen()
{
    // Random number assigned to i (1-25) and returned
    int i;
    srand((unsigned)time(0));
    i = (rand() % 25) + 1;
    return i;
}

/*
Description: Encrypts an upper case alphabetic character using the
             Caesar cipher
Input: upper case alphabetic character, valid key
Returns: encrypted version of ch
*/
char encrypt(char ch, int key)
{
    // Substitution shift for ASCII values of alphabet
    ch = ((ch - 65 + key) % 26) + 65;
    return ch;
}

/*
Description: Decrypts an upper case alphabetic character using the
             Caesar cipher
Input: upper case alphabetic character, valid key
Returns: decrypted version of input
*/
char decrypt (char ch, int key)
{
    // Substitution shift for ASCII values of alphabet
    ch = ((ch - 65 - key + 26) % 26) + 65;
    return ch;
}

/*
Description: function opens a file
Input: file stream object reference, name of the file, mode of open
Output: input file name is opened.
*/
void fileOpen(fstream& file, string name, char mode)
{
 string fileType;

 if (mode == 'r')
  fileType = "input";
```

```
 if (mode == 'w')
  fileType = "output";

 if (mode == 'r')
  file.open(name, ios::in);  //available thorugh fstream
 if (mode == 'w')
  file.open(name, ios::out);  //available through fstream;

 if (file.fail()) //error condition
 {
  cout << "Error opening " << fileType << " file" << endl;
  exit(EXIT_FAILURE);
 }
}
```