flipcode.h

```
#ifndef FLIPCODER_H
#define FLIPCODER_H

#include <QString>

class FlipCoder {
public:
    QString encode(const QString& input);
    QString decode(const QString& input);
};

#endif // FLIPCODER_H
```

flipcode.cpp

```
#include "flipcoder.h"

QString FlipCoder::encode(const QString& input) {
    QString encoded;
    for (const QChar& c : input) {
        ushort ascii = c.unicode();
        if (ascii >= 0x20 && ascii <= 0x7E && c != ' ') {
            encoded += QChar(0x7F - (ascii - 0x20));
        } else {
            encoded += c;
        }
    }
    return encoded;
}

QString FlipCoder::decode(const QString& input) {
    QString decoded;
    for (const QChar& c : input) {
        ushort ascii = c.unicode();
        if (ascii >= 0x20 && ascii <= 0x7E && c != ' ') {
            decoded += QChar(0x7F - (ascii - 0x20));
        } else {
            decoded += c;
        }
    }
    return decoded;
}
```

invertcasecoder.h

```
#ifndef INVERTCASECODER_H
#define INVERTCASECODER_H

#include <QString>
```

```
class InvertCaseCoder {
public:
    QString encode(const QString& input);
    QString decode(const QString& input);
};

#endif // INVERTCASECODER_H
```

<u>invertcasecoder.cpp</u>

```
#include "invertcasecoder.h"

QString InvertCaseCoder::encode(const QString& input) {
    QString encoded;
    for (const QChar& c : input) {
        if (c.isUpper()) {
            encoded += c.toLower();
        } else if (c.isLower()) {
            encoded += c.toUpper();
        } else {
            encoded += c;
        }
    }
    return encoded;
}

QString InvertCaseCoder::decode(const QString& input) {
    return encode(input); // Decoding is the same as encoding for invert case
}
```

<u>optcoder.h</u>

```
#ifndef OTPCODER_H
#define OTPCODER_H

#include <QString>

class OTPCoder {
public:
    OTPCoder();
    QString encode(const QString& input);
    QString decode(const QString& input);

private:
    int pad[11]; // Your pad values go here
};

#endif // OTPCODER_H
```

<u>optcoder.cpp</u>

```
#include "otpcoder.h"

OTPCoder::OTPCoder() {
    // Initialize your OTP pad with prime numbers
    pad[0] = 17;
    pad[1] = 29;
    pad[2] = 5;
    pad[3] = 12;
    pad[4] = 6;
    pad[5] = 22;
    pad[6] = 11;
    pad[7] = 30;
    pad[8] = 8;
    pad[9] = 18;
    pad[10] = 15;
}

QString OTPCoder::encode(const QString& input) {
    QString encoded;
    int padIndex = 0;

    for (const QChar& c : input) {
        ushort ascii = c.unicode();
        if (ascii >= 0x20 && ascii <= 0x7E) {
            int offset = pad[padIndex];
            padIndex = (padIndex + 1) % 11;

            int encodedAscii = ascii + offset;
            if (encodedAscii > 0x7E) {
                encodedAscii -= 0x5F; // Bring it back into the range 0x20 to 0x7E
            }
            encoded += QChar(encodedAscii);
        } else {
            encoded += c;
        }
    }
    return encoded;
}

QString OTPCoder::decode(const QString& input) {
    QString decoded;
    int padIndex = 0;

    for (const QChar& c : input) {
        ushort ascii = c.unicode();
        if (ascii >= 0x20 && ascii <= 0x7E) {
            int offset = pad[padIndex];
            padIndex = (padIndex + 1) % 11;

            int decodedAscii = ascii - offset;
            if (decodedAscii < 0x20) {
                decodedAscii += 0x5F; // Bring it back into the range 0x20 to 0x7E
            }
            decoded += QChar(decodedAscii);
        } else {
            decoded += c;
        }
    }
    return decoded;
}
```

main.cpp

```
#include <iostream>
#include <QString>
#include <string> // Include the <string> header for reading input as a std::string

#include "flipcoder.h"
#include "otpcoder.h"
#include "invertcasecoder.h"

void processInput(const std::string& input) {
    QString qInput = QString::fromStdString(input); // Convert std::string to QString

    // Display the original input
    std::cout << "Input: <" << qInput.length() << ">[" << qInput.toStdString() << "]\n";

    // Test FlipCoder
    FlipCoder flipCoder;
    QString flipped = flipCoder.encode(qInput);
    std::cout << "Flip Encoded: <" << flipped.length() << ">[" << flipped.toStdString() << "]\n";

    QString flippedDecoded = flipCoder.decode(flipped);
    std::cout << "Flip Decoded: <" << flippedDecoded.length() << ">[" <<
flippedDecoded.toStdString() << "]\n";

    // Test OTPCoder
    OTPCoder otpCoder;
    QString otpEncoded = otpCoder.encode(qInput);
    std::cout << "OTP Encoded: <" << otpEncoded.length() << ">[" <<
otpEncoded.toStdString() << "]\n";

    QString otpDecoded = otpCoder.decode(otpEncoded);
    std::cout << "OTP Decoded: <" << otpDecoded.length() << ">[" <<
otpDecoded.toStdString() << "]\n";
```

```
    // Test InvertCaseCoder
    InvertCaseCoder invertCaseCoder;
    QString invertCaseEncoded = invertCaseCoder.encode(qInput);
    std::cout << "Invert Case Encoded: <" << invertCaseEncoded.length() << ">[" <<
invertCaseEncoded.toStdString() << "]\n";

    QString invertCaseDecoded = invertCaseCoder.decode(invertCaseEncoded);
    std::cout << "Invert Case Decoded: <" << invertCaseDecoded.length() << ">[" <<
invertCaseDecoded.toStdString() << "]\n";
}


int main() {
    std::string input;
    std::cout << "Enter a text string (up to 1000 characters): ";
    std::getline(std::cin, input); // Read input as a std::string
    processInput(input);
    std::cout << "\n";
    input = "Hung-Ting Lee";
    processInput(input);
    std::cout << "\n";
    input = "In a hole in the ground there lived a hobbit. 1234567890~ !@#$%^&*()";
    processInput(input);
    std::cout << "\n";
    input = "Virginia Tech is a public university and one of Virginia's two land-grant
institutions";
    processInput(input);

    return 0;
}
```

<u>the console output</u>

```
Input: <13>[Hung-Ting Lee]
Flip Encoded: <13>[W*18rK618 S::]
Flip Decoded: <13>[Hung-Ting Lee]
OTP Encoded: <13>[Y3ss3jt-o2[v#]
OTP Decoded: <13>[Hung-Ting Lee]
Invert Case Encoded: <13>[hUNG-tING lEE]
Invert Case Decoded: <13>[Hung-Ting Lee]

Input: <68>[In a hole in the ground there lived a hobbit. 1234567890~ !@#$%^&*()]
Flip Encoded: <68>[V1 > 703: 61 +7: 8-0*1; +7:-: 36):; > 70==6+q nmlkjihgfo! ~_|{zAyuwv]
Flip Decoded: <68>[In a hole in the ground there lived a hobbit. 1234567890~ !@#$%^&*()]
OTP Encoded: <68>[Z,%m&~z+m2x
=ytk6r1w(}u=ytk)p>t{&v"%m&~z!j{$?=6>9J@T?JHA<%-F9/Cf899F]
OTP Decoded: <68>[In a hole in the ground there lived a hobbit. 1234567890~ !@#$%^&*()]
Invert Case Encoded: <68>[iN A HOLE IN THE GROUND THERE LIVED A HOBBIT.
1234567890~ !@#$%^&*()]
```

```
Invert Case Decoded: <68>[In a hole in the ground there lived a hobbit. 1234567890~
!@#$%^&*()]

Input: <86>[Virginia Tech is a public university and one of Virginia's two land-grant
institutions]
Flip Encoded: <86>[I6-8616> K:<7 6, > /*=36< *16):-,6+& >1; 01: 09 I6-8616>x, +(0
3>1;r8->1+ 61,+6+*+601,]
Flip Decoded: <86>[Virginia Tech is a public university and one of Virginia's two land-grant
institutions]
OTP Encoded: <86>[g'wso%t (ftt&%uy6l>x(q}'h,{%t5m%#z2~,g%o>w!t1-k,\
}&q!xrDx,z.z>ts}uJl~g% >q!#&'y"z z-{]
OTP Decoded: <86>[Virginia Tech is a public university and one of Virginia's two land-grant
institutions]
Invert Case Encoded: <86>[vIRGINIA tECH IS A PUBLIC UNIVERSITY AND ONE OF
vIRGINIA'S TWO LAND-GRANT INSTITUTIONS]
Invert Case Decoded: <86>[Virginia Tech is a public university and one of Virginia's two
land-grant institutions]
```