```
#ifndef EXCEPTION_H
#define EXCEPTION_H

#include <string>

using namespace std;

class Exception {
protected:
   string error;
public:
   Exception(string& error);
   string getError();
};

#endif
```

```
#ifndef ACCESSDEVICE_H
#define ACCESSDEVICE_H

#include "Exception.h"

#include <string>

using namespace std;

class AccessDevice {
public:
   virtual string registerStudent(const string& s) = 0;
   virtual string authenticateStudent(const string& s) = 0;
};

#endif
```

```
#ifndef CONTACTLESSDEVICE_H
#define CONTACTLESSDEVICE_H

#include "AccessDevice.h"
#include <string>


class ContactlessDevice : public AccessDevice {

protected:
   int length = 0;
   int * keyword;
   int maxStudent;
   int currentStudent;
   int ** rotor;
   string ** studentDatabase;

public:
   string registerStudent(const string& s);
   string authenticateStudent(const string& s);
   int * setKeyword(const string& s);
   void setRotor();
   int ** getRotor();
   string ** getStudentDatabase();

protected:
   virtual char hashChar(char) = 0;


};

#endif
```

```cpp
#include "ContactlessDevice.h"

#include <string>
#include <sstream>
#include <iostream>
#include "Exception.h"

using namespace std;

string ContactlessDevice::registerStudent(const string& s) {
    string e = "Student Database is full";
    if (this->maxStudent == this->currentStudent) {throw Exception(e);}

    string out = "";
    this->studentDatabase[currentStudent][0] = s;
    for (int i = 0; i < s.length(); i++) {
        out += to_string(hashChar(s[i]));
    }

    e = "Student already exist";
    for (int i = 0; i < this->maxStudent; i++) {
        if (this->studentDatabase[i][0] == s && this->studentDatabase[i][1] == out) {
            throw Exception(e);
        }
    }

    studentDatabase[currentStudent][1] = out;
    this->currentStudent++;

    return out;
}

string  ContactlessDevice::authenticateStudent(const string& s) {
    string out = "";
    for (int i = 0; i < s.length(); i++) {
        out += to_string(hashChar(s[i]));
    }
    string flag = "false";
    for (int i = 0; i < this->maxStudent; i++) {
        if (this->studentDatabase[i][0] == s && this->studentDatabase[i][1] == out) {
            flag = "true";
```



```cpp
            ss >> this->keyword[i];
            ss.clear();
        }

        return this->keyword;
}

int ** ContactlessDevice::getRotor() {
    return this->rotor;
}

void ContactlessDevice::setRotor() {
    string e = "Keyword must be set";
    if (keyword == nullptr) {
        throw Exception(e);
    }

    //Set Rotor:
    rotor = new int*[10];
    for (int i = 0; i < 10; i++) {
        rotor[i] = new int[length];
    }

    int counter = 0;
    for (int i = keyword[0]; i < keyword[0] + 10; i++) {
        rotor[counter][0] = i;
        if (rotor[counter][0] >= 10) {rotor[counter][0] -= 10;}
        counter++;
    }

    for (int i = 0; i < 10; i++) {
        for (int j = 1; j < length; j++) {
            rotor[i][j] = keyword[j] + rotor[i][j - 1];
            if (rotor[i][j] >= 10) {rotor[i][j] -= 10;}
        }
    }
}

string ** ContactlessDevice::getStudentDatabase() {
    return this->studentDatabase;
}
```

```
#ifndef PALMVEIN_H
#define PALMVEIN_H

#include "ContactlessDevice.h"

#include <string>

using namespace std;

class PalmVein : public ContactlessDevice {
public:
   PalmVein(const string& s, const int& maxStud);
   ~PalmVein();
   //string registerStudent(const string&);
   //string authenticateStudent(const string&);

protected:
   char hashChar(char c);
};

#endif
```

```cpp
#include "PalmVein.h"
#include "ContactlessDevice.h"

#include <string>
#include <sstream>
#include <iostream>

using namespace std;

PalmVein::PalmVein(const string &s, const int &maxStud) {
   this->studentDatabase = nullptr;
   this->rotor = nullptr;

   string e = "The keyword provided is not going to generate a safe encryption";
   if (s.length() <= 1) { throw Exception(e); }

   this->studentDatabase = new string *[maxStud];
   for (int i = 0; i < maxStud; i++) {
      this->studentDatabase[i] = new string[2];
      this->studentDatabase[i][0] = "z";
      this->studentDatabase[i][1] = "z";
   }
   this->currentStudent = 0;
   this->maxStudent = maxStud;
   this->setKeyword(s);
}

PalmVein::~PalmVein() {
   //Keyword:
   delete [] this->keyword;

   //**rotor:
   if (this->rotor != nullptr) {
      for (int i = 0; i < 10; i++) {
         delete this->rotor[i];
      }
      delete [] this->rotor;
   }

   //studentDatabase:
   if (this->studentDatabase != nullptr) {
```



```cpp
   return out;
}
```

```
#ifndef FACIALRECOGNITION_H
#define FACIALRECOGNITION_H

#include "ContactlessDevice.h"

class FacialRecognition : public ContactlessDevice {
private:
   int stepSize;

public:
   FacialRecognition(const string &k, const int &max, int stepSize);
   ~FacialRecognition();
   void setStepSize(int step);
   string registerStudent(const string &s);
   string authenticateStudent(const string &s);

protected:
   char hashChar(char);
};

#endif
```

```cpp
#include "FacialRecognition.h"
#include "ContactlessDevice.h"

#include <iostream>
#include <sstream>

using namespace std;

FacialRecognition::FacialRecognition(const string &k, const int &max, int stepSize) {
    this->setKeyword(k);
    this->maxStudent = max;
    this->stepSize = stepSize;
    this->currentStudent = 0;
    this->studentDatabase = new string*[max];
    for (int i = 0; i < max; i++) {
        this->studentDatabase[i] = new string[2];
        this->studentDatabase[i][0] = "z";
        this->studentDatabase[i][1] = "z";
    }
    this->rotor = nullptr;
}

FacialRecognition::~FacialRecognition() {
    //Keyword:
    delete [] this->keyword;

    //**rotor:
    if (this->rotor != nullptr) {
        for (int i = 0; i < 10; i++) {
            delete this->rotor[i];
        }
        delete [] this->rotor;
    }

    //studentDatabase:
    if (this->studentDatabase != nullptr) {
        for (int i = 0; i < this->maxStudent; i++) {
            delete [] this->studentDatabase[i];
        }
        delete [] this->studentDatabase;
    }
```



```cpp
    //Save Rotor & keyword:
    int * tempKey = new int[length];
    for (int i = 0; i < length; i++) {tempKey[i] = this->keyword[i];}

    std::string out = ContactlessDevice::authenticateStudent(s);

    //Restore rotor & keyword:
    for (int i = 0; i < length; i++) {
        this->keyword[i] = tempKey[i];
    }
    this->setRotor();
    delete tempKey;

    return out;
}

char FacialRecognition::hashChar(char c) {
    /*cout << endl;
    for (int i = 0; i < 10; i++) {
        for (int j = 0; j < length; j++) {
            cout << this->rotor[i][j] << "\t";
        }
        cout << endl;
    }
    cout << endl;*/

    int temp;
    stringstream ss;
    ss << c;
    ss >> temp;
    int out;
    for (int i = 0; i < 10; i++) {
        if (this->rotor[i][length - 1] == temp) {
            out = i;
        }
    }

    //Rotate Table:
    for (int i = 0; i < 10; i++) {
        //store:
```