```
//////////////////////////////////////////////////////////
// Grade.idl, Troy


module Grade {
  interface Grader {
    boolean add_grade(in string tid, in string pwd, in float grade);
    float show_grade(in string sid, in string pwd);
  };
  interface Security {
    boolean check_teacher_pwd(in string tid, in string pwd);
    boolean check_student_pwd(in string sid, in string pwd);
  };
};
```

```
////////////////////////////////////////////////////////
// Grader.C, Troy

#include "GradeImpl.h"

USE_STD_NS
int main(int argc, char* const* argv)
{
  try {
    CORBA::ORB_var orb = CORBA::ORB_init(argc, argv);

    CORBA::Object_var obj = orb->resolve_initial_references("RootPOA");
    PortableServer::POA_var rootPOA = PortableServer::POA::_narrow(obj);

    CORBA::PolicyList policies;
    policies.length(1);
    policies[(CORBA::ULong)0] = rootPOA->create_lifespan_policy(PortableServer::PERSISTENT);

    PortableServer::POAManager_var poa_manager = rootPOA->the_POAManager();
    PortableServer::POA_var myPOA = rootPOA->create_POA("zzhan_grader_poa",
                                                        poa_manager,
                                                        policies);
    GraderImpl graderServant;

    PortableServer::ObjectId_var graderId =
      PortableServer::string_to_ObjectId("zzhan_Grader");
    myPOA->activate_object_with_id(graderId, &graderServant);

    poa_manager->activate();

    CORBA::Object_var reference = myPOA->servant_to_reference(&graderServant);
    cout << reference << " is ready" << endl;

    orb->run();
  }catch(const CORBA::Exception& e) {
    cerr << e << endl;
    return 1;
  }
  return 0;
}
```

```
////////////////////////////////////////////////////////
// Security.C, Troy

#include "GradeImpl.h"

USE_STD_NS
int main(int argc, char* const* argv)
{
  try {

    CORBA::ORB_var orb = CORBA::ORB_init(argc, argv);

    CORBA::Object_var obj = orb->resolve_initial_references("RootPOA");
    PortableServer::POA_var rootPOA = PortableServer::POA::_narrow(obj);

    CORBA::PolicyList policies;
    policies.length(1);
    policies[(CORBA::ULong)0] = rootPOA->create_lifespan_policy(PortableServer::PERSISTENT);

    PortableServer::POAManager_var poa_manager = rootPOA->the_POAManager();
    PortableServer::POA_var myPOA = rootPOA->create_POA("zzhan_security_poa",
                                                        poa_manager,
                                                        policies);

    SecurityImpl securityServant;
    PortableServer::ObjectId_var securityId =
                           PortableServer::string_to_ObjectId("zzhan_Security");
    myPOA->activate_object_with_id(securityId, &securityServant);
    poa_manager->activate();

    CORBA::Object_var reference = myPOA->servant_to_reference(&securityServant);
    cout << reference << " is ready" << endl;
    orb->run();
  }catch(const CORBA::Exception& e) {
    cerr << e << endl;
    return 1;
  }
  return 0;
}
```

```
/////////////////////////////////////////////////////////
// Student.C, Troy

#include "Grade_c.hh"

USE_STD_NS

int main(int argc, char* const* argv)
{
  try {

    CORBA::ORB_var orb = CORBA::ORB_init(argc, argv);
    PortableServer::ObjectId_var graderId =
      PortableServer::string_to_ObjectId("zzhan_Grader");

    Grade::Grader_var grader =
      Grade::Grader::_bind("/zzhan_grader_poa", graderId);

    const char* sid = argc > 1 ? argv[1] : "1001";
    const char* pwd = argc > 2 ? argv[2] : "test";

    CORBA::Float grade = grader->show_grade(sid,pwd);
    if (grade<0)
      cout<<"Student(CPP): Invalid SID or Password"<<endl;
    else
      cout << "Student(CPP): The grade of: "<<sid<<" is: "<<grade<<endl;
  }catch(const CORBA::Exception& e) {
    cerr << e << endl;
    return 1;
  }
  return 0;
}
```

```
////////////////////////////////////////////////////////
// Teacher.C, Troy

#include "Grade_c.hh"
#include <stdlib.h>
#include <math.h>

USE_STD_NS
int main(int argc, char* const* argv)
{
  try {
    CORBA::ORB_var orb = CORBA::ORB_init(argc, argv);
    PortableServer::ObjectId_var graderId =
                        PortableServer::string_to_ObjectId("zzhan_Grader");

    Grade::Grader_var grader =
                  Grade::Grader::_bind("/zzhan_grader_poa", graderId);

    const char* tid = argc > 1 ? argv[1] : "9999";
    const char* pwd = argc > 2 ? argv[2] : "test";
    const char* str_grade = argc > 3 ? argv[3] : "85.50";
    float grade=(float)atof(str_grade);

    CORBA::Boolean ret = grader->add_grade(tid,pwd,grade);
    if (!ret)
      cout<<"Teacher(CPP): Invalid TID or Password."<<endl;
    else
      cout << "Teacher(CPP): The grade is updated to: "<<grade<<endl;
  }catch(const CORBA::Exception& e) {
    cerr << e << endl;
    return 1;
  }
  return 0;
}
```

```
//////////////////////////////////////////////////////////
// GradeImpl.h, Troy

#include "Grade_s.hh"
#include <math.h>
#include <string.h>

USE_STD_NS
class GraderImpl : public virtual POA_Grade::Grader{
 private:
  CORBA::Float _grade;
 public:
  GraderImpl():_grade(0.0){}

  CORBA::Boolean add_grade(const char* tid, const char *pwd, CORBA::Float grade){
    CORBA::Boolean ret=false;
    try {
      PortableServer::ObjectId_var securityId =
        PortableServer::string_to_ObjectId("zzhan_Security");

      Grade::Security_var security =
        Grade::Security::_bind("/zzhan_security_poa", securityId);

      CORBA::Boolean result = security->check_teacher_pwd(tid,pwd);
      if (!result)
        cout<<"Grader(CPP): Invalid TID or Password."<<endl;
      else{
        cout << "Grader(CPP): Teacher: "<<tid<<" logged in. The grade is updated to: "<<grade<<endl;
        _grade=grade;
        ret=true;
      }
    }
    catch(const CORBA::Exception& e) {
      cerr << e << endl;
      return false;
    }
    return ret;
  }

  CORBA::Float show_grade(const char* sid, const char *pwd) {
    CORBA::Float ret=-1.0;
    try {
      PortableServer::ObjectId_var securityId =
        PortableServer::string_to_ObjectId("zzhan_Security");
      Grade::Security_var security =
        Grade::Security::_bind("/zzhan_security_poa", securityId);
      CORBA::Boolean result = security->check_student_pwd(sid,pwd);
      if (!result)
        cout<<"Grader(CPP): Invalid TID or Password."<<endl;
      else{
        cout << "Grader(CPP): Student: "<<sid<<" logged in. The grade: "<<_grade<<" is retrieved."<<endl;
        ret= _grade;
      }
    }catch(const CORBA::Exception& e) {
      cerr << e << endl;
      return -1.0;
    }
    return ret;
  }
};

#define TROY_MAX 100
class SecurityImpl : public POA_Grade::Security{
private:
  char* _tid[TROY_MAX];
  char* _t_pwd[TROY_MAX];
  int _t_length;

  char* _sid[TROY_MAX];
  char* _s_pwd[TROY_MAX];
  int _s_length;
public:
  SecurityImpl(){
    _tid[0]="9999";
    _t_pwd[0]="test";
    _tid[1]="8888";
    _t_pwd[1]="test";
    _tid[2]="7777";
    _t_pwd[2]="test";
    _t_length=2;

    _sid[0]="1001";
    _s_pwd[0]="test";
    _sid[1]="2002";
    _s_pwd[1]="test";
    _sid[2]="3003";
    _s_pwd[2]="test";
    _s_length=2;
  }

  CORBA::Boolean check_teacher_pwd(const char* tid, const char* pwd) {
    int i=0,j=0;
    int found=0;
    while ((i<=_t_length)&&(!found)){
      if ((j=strcmp(_tid[i],tid))==0)
        found=1;
      else
        i++;
    }
    if(i>_t_length){
      _tid[i]=(char *)malloc(sizeof(char)*strlen(tid));
      strcpy(_tid[i],tid);
      _t_pwd[i]=(char *)malloc(sizeof(char)*strlen(pwd));
      strcpy(_t_pwd[i],pwd);
      _t_length++;
      cout<<"Security(CPP): Created TID "<<tid<<" : "<<pwd<<endl;
      return 1;
    }
    if((j=strcmp(_t_pwd[i],pwd))==0){
      cout<<"Security(CPP): Authentication succeeded on TID:"<<tid<<endl;
      return 1;
    }else{
      cout<<"Security(CPP): Authentication failed on TID:"<<tid<<endl;
    }
    return 0;
  }

  CORBA::Boolean check_student_pwd(const char* sid, const char* pwd) {
    int i=0,j=0;
    int found=0;
    while ((i<=_s_length)&&(!found)){
      if ((j=strcmp(_sid[i],sid))==0)
        found=1;
      else
        i++;
    }
    if(i>_s_length){
      _sid[i]=(char *)malloc(sizeof(char)*strlen(sid));
      strcpy(_sid[i],sid);
      _s_pwd[i]=(char *)malloc(sizeof(char)*strlen(pwd));
      strcpy(_s_pwd[i],pwd);
      _s_length++;
      cout<<"Security(CPP): Created SID "<<sid<<" : "<<pwd<<endl;
      return 1;
    }
    if((j=strcmp(_s_pwd[i],pwd))==0){
      cout<<"Security(CPP): Authentication succeeded on SID:"<<sid<<endl;
      return 1;
    }else{
      cout<<"Security(CPP): Authentication failed on SID:"<<sid<<endl;
    }
    return 0;
  }

};
```