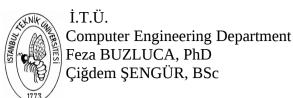
18.06.2001



## **Object Oriented Programming Final Examination Solutions**

```
QUESTION 1:
  * 0001f1.cpp
    2000-2001 Final Exam
                               Answer 1**/
#include <iostream>
#include <string>
using namespace std;
class Person{
  string name;
  short int age;
 public:
  Person(){ name=""; age=0;}
  Person(const string &,int);
  bool operator < (const Person &) const; // for comapring</pre>
  bool operator == (const Person &) const; // for searching
  void operator () () const;
};
Person::Person(const string & n, int a)
  name=n;
  age=a;
}
bool Person::operator < (const Person &p) const</pre>
                                    // if not empty
  if (age) return age<p.age;
     else return false;
}
bool Person::operator == (const Person &p) const
  return (name == p.name && age==p.age);
void Person::operator () () const
  cout << "Name: " << name <<" Age: " << age;
template <class Type>
class Array{
   private:
    Type *elem;
    int size;
   public:
                                               // constructor
    Array(int);
    Type & operator[](int);
     bool find(const Type& ) const;
     const Type& smallest() const;
    ~Array(){delete [] elem;}
                                                // destructor
```

};

```
template<class Type>
Array<Type>::Array(int s)
  size=s;
  elem=new Type[size];
}
template<class Type>
Type & Array<Type>::operator[](int i)
   if(i<0 || i>=size)
        throw "Out of bounds!"; // throw exception
   return elem[i];
template<class Type>
bool Array<Type>::find(const Type& ser) const
  for (int i=0 ; i<size; i++)
      if (elem[i]== ser) return true;
  return false;
}
template<class Type>
const Type & Array<Type>::smallest() const
{
  int i=0;
  for (int j=1 ; j<size; j++)
    if (elem[j] < elem[i]) i=j;</pre>
  return elem[i];
}
QUESTION 2:
  ** 0001f2.cpp
    2000-2001 Final Exam Answer 2 **/
#include <iostream>
#include <string>
using namespace std;
                                                           // person class
class person
{
  protected:
   string name;
  public:
   person(){
                                                           // Constructor
       cout << " Enter name: ";
       cin >> name;
   virtual void print() const{
   cout << endl << "Name = " << name;}</pre>
   virtual bool isSuccessful()const =0; // Pure virtual (abstract class)
   virtual ~person(){}
};
                                                    // student class
class student : virtual public person
{
   float gpa;
                                                    // grade point average
  public:
                                                    // Constructor
    student(){
       cout << " Enter student's GPA: ";
```

```
cin >> gpa;
   }
   void print() const{
       person::print();
cout << endl << " GPA = " << gpa;</pre>
   bool isSuccessful() const{
       return (gpa > 3.5);
    virtual ~student(){}
};
                                           // teacher class
class teacher : virtual public person
  protected:
                                             // number of papers published
   int numPubs;
  public:
   teacher(){
                                            // Constructor
       cout << " Enter number of teacher's publications: ";</pre>
       cin >> numPubs;
   void print() const{
       person::print();
       cout << endl << " Publications = " << numPubs;</pre>
    bool isSuccessful() const{
       return (numPubs > 50);
   virtual ~teacher(){}
};
class assistant : public student , public teacher // assistant class
                                             // number of courses
   int numCourse;
  public:
                                            // Constructor
   assistant(){
       cout << " Enter number of courses: ";</pre>
       cin >> numCourse;
   void print() const{
       student::print();
       cout << endl << " Publications = " << numPubs;</pre>
       cout << endl << " Num. of Courses = " << numCourse;</pre>
    bool isSuccessful() const{
      return ( student::isSuccessful() && numCourse > 3) ;
   virtual ~assistant(){}
};
```

```
void main()
   person* persPtr[100];
                                   // list of pointers to persons
   int n = 0;
                                   // number of persons on list
   char choice;
                                   // 's', 't' or 'a'
   do{
      cout << "Enter student or teacher (s/t/a): ";
      cin >> choice;
      switch(choice) {
      case 's': persPtr[n++] = new student;
                                                    // put new student
                                                    //
                 break;
                                                       in array
                                                    // put new teacher
      case 't': persPtr[n++] = new teacher;
                                                    //
                                                       in array
                 break;
      case 'a': persPtr[n++] = new assistant;
                                                    // put new assistant
                                                    //
                                                       in array
      cout << " Enter another (y/n)? ";
                                                    // do another person?
   cin >> choice;
} while( (choice=='y') && (n<100) );</pre>
                                                    // cycle until not 'v'
   for(int j=0; j<n; j++)
                                                    // print names of all
      persPtr[j]->print();
                                                    // persons, and
      if (persPtr[j]->isSuccessful())
  cout << "\n (This person is succesfull)"; // say if succesfull</pre>
      delete persPtr[j];
                                                        // delete object
     }
}
                                                        // end main()
```