İ.T.Ü. 18.06.2001
Computer Engineering Department
Feza BUZLUCA, PhD
Çiğdem ŞENGÜR, BSc

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
   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
{
  if (age) return age<p.age;         // if not empty
     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:
     Array(int);                                  // constructor
     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;
  return elem[i];
}
```

QUESTION 2:

```
/** 0001f2.cpp
    2000-2001 Final Exam Answer 2 **/
#include <iostream>
#include <string>
using namespace std;

class person                                          // person class
{
  protected:
    string name;
  public:
    person(){                                         // Constructor
        cout << "  Enter name: ";
        cin >> name;
    }
    virtual void print() const{
        cout << endl << "Name = " << name;}
    virtual bool isSuccessful()const =0;   // Pure virtual (abstract class)
    virtual ~person(){}
};

class student : virtual public person           // student class
{
    float gpa;                                  // grade point average
  public:
    student(){                                  // Constructor
        cout << "  Enter student's GPA: ";
```

```
        cin >> gpa;
    }
    void print() const{
        person::print();
        cout << endl << "  GPA = " << gpa;
    }
    bool isSuccessful() const{
        return (gpa > 3.5);
    }
    virtual ~student(){}
};

class teacher : virtual public person     // teacher class
{
  protected:
    int numPubs;                          // number of papers published
  public:
    teacher(){                            // Constructor
        cout << "  Enter number of teacher's publications: ";
        cin >> numPubs;
    }
    void print() const{
        person::print();
        cout << endl << "  Publications = " << numPubs;
    }
    bool isSuccessful() const{
        return (numPubs > 50) ;
    }
    virtual ~teacher(){}
};

class assistant : public student , public teacher // assistant class
{
    int numCourse;                        // number of courses
  public:
    assistant(){                          // Constructor
        cout << "  Enter number of courses: ";
        cin >> numCourse;
    }
    void print() const{
        student::print();
        cout << endl << "  Publications = " << numPubs;
        cout << endl << "  Num. of Courses = " << numCourse;
    }
    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
      case 't': persPtr[n++] = new teacher;     // put new teacher
                break;                          //    in array
      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) );        // cycle until not 'y'
    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
      delete persPtr[j];                            // delete object
     }
}                                                   // end main()
```