

CSI2372A Advanced Programming Concepts with C++

Selected Question from the MIDTERM EXAMINATION 2006

Professor: Jochen Lang

Page 1 of 9

Family Name: _____

Other Names: _____

Student Number: _____

Signature _____

This exam is CLOSED BOOK.

Please answer the questions in this booklet.

If you do not understand a question, clearly state an assumption and proceed. You may answer a question in this set of questions or in an exam booklet.

No calculators or other electronic devices are allowed.

At the end of the exam, when time is up:

- Stop working and turn your exam upside down.
- Remain silent. Do not move or speak until all exams have been picked up, and a TA or the Professor gives the go-ahead to leave.

PART A: SHORT QUESTIONS (12 MARKS)

1. Consider the classes Furniture, Chair and Seat defined as follows:

```
class Furniture {
    float d_price;
public:
    Furniture(float _price)
        : d_price(_price)
    {}
    float getPrice() {
        return d_price;
    }
protected:
    void setPrice(float _price) {
        d_price = _price;
    }
};

class Chair : public Furniture {
protected:
    double d_height;
public:
    Chair( double _price,
           double _height = 0.65 )
        : Furniture( _price ),
          d_height( _height )
    {}
    double getHeight() {
        return d_height;
    }
    virtual void
    setHeight( double _height ) {
        d_height = _height;
    }
};
```

```
class Seat : public Chair {
    double d_cushion;
public:
    Seat( float _price,
          double _cushion = 10.0 )
        : Chair( _price ),
          d_cushion( _cushion ) {
        updateHeight();
    }
    float getPrice() {
        return 1.5f * Chair::getPrice();
    }
    void setHeight( double _height ) {
        d_height = _height;
        updateHeight();
    }
private:
    void updateHeight( ) {
        d_height += d_cushion;
    }
};

#include <iostream>

using std::cout;
using std::endl;

int main() {
    Seat soft( 100.0f );
    cout << "Price: "
         << soft.getPrice() << endl;
    Chair *hard = &soft;
    cout << "Price: "
         << hard->getPrice() << endl;
    hard->setHeight( 90.0 );
    cout << "Seat height: "
         << hard->getHeight() << endl;
    return 0;
}
```

Consider the main program:

- a. What will be printed by the main program? [3]
- b. Assume we add the following lines to the main routine (right before the return statement):

```
Furniture *diningSet = &soft;
```

```
diningSet.setPrice( 150.0f );
```

Find the error in the above lines of code. Briefly state the reason for the error [1].

- c. Assume we add the following lines to the main routine (right before the return statement):

```
Furniture *diningSet = &soft;  
Chair *chair = diningSet;  
chair->setHeight(75);
```

Find the error in the above lines of code. Briefly state the reason for the error [1].

- d. Find the error in the following lines of code. Briefly state the reason for the error [1].

```
Seat set[4];  
for ( int i=0; i<4; ++i ) {  
    cout << "Price: " << set[i].getPrice() << endl;  
}
```

2. What does the following program print [3]?

```
#include <iostream>

using std::cout;
using std::endl;

int foo( int _iVal ) {
    return 2*_iVal;
}

char foo( char _cVal ) {
    return _cVal - 2;
}

double foo( double _fVal ) {
    return _fVal/2.0f;
}

int main () {
    int iVal = 10;
    double dVal = 2.5;
    char cVal = 5;
    short sVal = 20;

    cout << cVal%sVal << endl;
    cout << iVal/dVal << endl;
    dVal += iVal/sVal * cVal;
    cout << dVal << endl;
    cout << foo( sVal) << endl;
    dVal = 2.5;
    cout << foo( cVal * dVal * 2 ) << endl;
    cout << foo( static_cast<double>(cVal) * foo(foo(cVal)+1)) << endl;

    return 0;
}
```

3. Consider the following program:

```
#include <iostream>

using std::cout;
using std::endl;

void printArray( double _dValArr[][2] ) {
    for ( int r=0; r<3; ++r ) {
        for ( int c=0; c<2; ++c ) {
            cout << _dValArr[r][c] << " ";
        }
        cout << endl;
    }
    cout << endl;
    return;
}

int main () {
    double dValArr[][2] = { 1.5, 2.5, 3.5, 4.5, 5.5, 6.5 };
    double *dValPtr = &dValArr[0][0];

    *(dValPtr+2) -= 3.0;
    printArray( dValArr );                                // -1-

    double (*dPtr)[2] = &dValArr[2];
    (*dPtr)[0] -= 5.0;
    printArray( dValArr );                                // -2-

    cout << (&dValArr[2][1] - dValPtr + 1) << endl; // -3-

    return 0;
}
```

a. What is printed in line -1- ? [1].

b. What is printed in line -2- ? [1].

c. What is printed in line -3- ? [1].

PART B: PROGRAMMING QUESTIONS (10 MARKS)

1. A class Stack for type double

- Class definition in stack.hh

```
static const double NaN;

class Stack {
    int d_capacity;
    int d_size;
    double *d_tab;

public:
    // Constructors
    Stack(int _capacity);
    Stack(const Stack &_oStack);

    // Put an element on to the stack
    bool push(double _element);

    // Retrieve an element from the stack
    double pop();

    // Destructor
    ~Stack();
};
```

- a. Define a 1 argument constructor with the capacity of the stack as argument.
The capacity of the stack is the maximum number of elements which the stack can hold. [2]

- b. Define the method `push`. The method should return `false` if the stack is full and the element can not be placed on the stack. It should return on success. [2]

- c. Define the method `pop` returning the last number inserted into `Stack`. The method should return `NaN` if the stack is empty [2].

- d. Define the copy constructor for the class `Stack` [4].