**COEN 244 - Winter 2012/13**

**Final Exam**

**~ Open Textbook & Course Notes ~**

**Name: ID:**

***Write your answers within the empty boxes – anything written outside these regions will NOT be marked. The questions are all equally weighted. Best wishes*☺**

**Q1**. Will the following program compile without errors? If it compiles and runs then what will it output?

**Y/N: N**

**Output:**

// classes example

#include <iostream>

using namespace std;

class CTriangle {

 float x, y;

 protected:

 void set\_values (float,float);

 float area () {return ( (x\*y)/2) );}

};

void CTriangle::set\_values (float a, float b) {

 x = a;

 y = b;

}

int main () {

 CTriangle tri;

 tri.set\_values (5.0,5.0);

 cout << "area: " << tri.area();

 return 0;

}

**Q2**. Is there an error in the following class? If there is an error, copy the line that has it.

**Y/N: N**

**Error:**

class Derived: public Base

{

public:

    double m\_dValue;

    Derived(double dValue=0.0, int nValue=0)

        : Base(nValue), m\_dValue(dValue)

    {}

};

// Assume that class Base has a general constructor that uses the single argument

// passed to it (through nValue), to initialize a private data member of type int.

**Q3.** Does the following class have a default constructor? If it does, what is the type of that constructor?

**Yes/No: Yes**

**Type: Default (Hidden)**

class arithmatics {

 public:

 int a,b,c;

 void add (int n, int m) { a=n; b=m; c=a+b; }

 };

**Q4**. Complete the following table, which summarizes the various levels of access that members of a class, members of a derived class (from that class) and a public user of an object instantiated from the class has to members of the (same) class. Assume public inheritance is used to create the derived class from the (original) class.

|  |  |  |  |
| --- | --- | --- | --- |
| **Access** | **public** | **protected** | **private** |
| Same class | yes | yes | \_yes\_\_ |
| Derived classes | yes | \_yes\_ | \_no\_ |
| Public (object) | yes | \_no\_ | no |

**Q5**. What data member must Square class have, and what could its access specifier be, in order for derived class Cube to be correct?

**Member: float length**

**Access: protected (or public)**

class Cube: protected Square

{ public: float volume () { return (length\*length\*length) ; } };

**Q6&7**. What is the output of the following program?

**Value of A[2] : 2**

**Value of A[5] : 5**

**Index out of bounds**

**Value of A[12] : 0**

#include <iostream>

using namespace std;

const int SIZE = 10;

class safearay

{

 private:

 int arr[SIZE];

 public:

 safearay()

 {

 register int i;

 for(i = 0; i < SIZE; i++)

 {

 arr[i] = i;

 }

 }

 int &operator[](int i)

 {

 if( i > SIZE )

 {

 cout << "Index out of bounds" <<endl;

 // return first element.

 return arr[0];

 }

 return arr[i];

 }

};

int main()

{

 safearay A;

 cout << "Value of A[2] : " << A[2] <<endl;

 cout << "Value of A[5] : " << A[5]<<endl;

 cout << "Value of A[12] : " << A[12]<<endl;

 return 0;

}

**Q8**. Is there a syntax error in the following code? If so, copy the line that contains the error?

int divide(int a, int b=2)

**Yes/No: Yes**

**Error:
int divide (int a, int b=2)**

{

 int r; r=a/b; return (r);

}

int divide(int a)

{

 int r=a/1; return (r);

}

**Q9.** Is the following class a **composition** or an **aggregation** or **neither**?

**Composition**

class PersonalComputer

{

private:

   int m\_cCPU;

   float m\_cMotherboard;

   string m\_cRAM;

};

**Q10&11**. The following program is missing one necessary statement from the main() function: what is the statement? And, at what numbered position could this statement be inserted to ensure correct functioning?

// virtual members

**Missing Statement:**

**ppoly1->set\_values(x, y);**

**Insertion Location: 30**

#include <iostream>

using namespace std;

class CPolygon {

 protected:

 int width, height;

 public:

 void set\_values (int a, int b)

 {width=a; height=b;}

 virtual int area ()

 { return (0); }

 };

class CRectangle: public CPolygon {

 public:

 int area ()

 { return (width \* height); }

 };

class CSquare: public CPolygon {

 public:

 int area ()

 { return (width \* width); }

 };

int main () {

 //10

 CRectangle rect;

 //20

 CPolygon \* ppoly1 = &rect;

 //30

 cout << ppoly1->area() << endl;

 //40

 return 0;

}

**Q12.** Is there an error in any of the following statements? If yes, re-write the statement to make it correct.

**Yes/No: Yes**

**Correct Version:
tommy = &number;**

int number;

int \*tommy;

\*tommy = number;

**Q13**. If a behavior of a base class, embodied as a member function of that base class, is a common behavior to almost all existing and potential sub-classes of the base class, then would you make that function: **virtual**, **pure virtual** or **neither**?

**virtual**

**Q14&15**. Add the necessary code to the following program in order to make it into a full and correct program capable of the output listed below it. Mark the **location** of the added code with an arrow (🡪).

class CPolygon {

**New Code:
int area() { return 0.5 \* width \* height; }**

 protected:

 int width, height;

 public:

 void set\_values (int a, int b)

 { width=a; height=b; }

 virtual int area ()= 0;

 void printarea (void)

 { cout << this->area() << endl; }

}; ↓

class CTriangle: public CPolygon { };

int main() {CPolygon\* ppoly = new CTriangle; ppoly->set\_values (4,5);

ppoly->printarea(); delete ppoly; return 0;}

// Output: 10

**Q16**. Is it possible to write one function template that satisfies all of the following function calls?

int nValue = min(3, 7);

**Yes/No:
No (C++89)
Yes (C++11)**

int nValue = min(3, 7, 9);

double dValue = min(6.34, 18.523, 23.44);

**Q17**. What is syntactically wrong with the following function? Simply copy/write the wrong/missing **line** in the box.

**template <typename T>**

T KAverage(T \*atArray, int nNumValues)

{

    T tSum = 0;

    for (int nCount= --nNumValues; nCount>0; nCount--)

        tSum += atArray[nCount];

    tSum /= nNumValues;

    return tSum;

}

**Q18**. If you needed to build a container class that would work equally well for integer, float and string arrays holding different numbers of elements, what kind of class would you build?

**Template container class**

**Q19&20**. If you wanted to record statistics about a class (e.g., number of created and deleted objects), you can store such values in global variables independent of the class or specially marked objects of the class, but it is best that this is done using of the class. Also, if hello() is a public static member function of

 **Static members**

class world, could it be called using world.hello()?

 **Yes/No: No**

**Q21**. What is the name of the mechanism used below for error handling? What will the following program generate in output (if any)?

**Name: Exception Handling**

**Output: Result=0**

#include <iostream>

using namespace std;

double division(int a, int b)

{

 if( b == 0 )

 {

 throw "Division by zero condition!";

 }

 return (a/b);

}

int main ()

{

 int x = 0;

 int y = 10;

 double z = 10.0;

 try {

 z = division(x, y);

 cout << "Result=" << z << endl;

 }

 catch (const char\* msg) {

 cerr << msg << endl;

 }

 catch (int value){

 cerr << "Division by zero condition because of " << value << endl;

 }

 catch (…) {cerr << "all OK" << endl;}

 return 0;

}

**Q22&Q23.**

Can more than one class be derived from the CVector class?

 **Yes/No: Yes**

Can an object be instantiated from the CVector class?

 **Yes/No: No**

class CVector {

 public:

 int x,y;

 CVector () {};

 CVector (int,int);

 virtual CVector operator+(CVector) = 0;

};

**Q24.** Does the following program compile and run? If yes, what is the output?

#include <vector>

**Yes/No: Yes Output: 0 1 2 3 4**

**Return value is optional (compiler-dependent)**

#include <iostream>

int main()

{

    using namespace std;

    vector<int> vect;

    for (int nCount = 5; nCount > 0; nCount--)

        vect.push\_back(5 - nCount);

    for (int nIndex=0; nIndex < vect.size(); nIndex++)

        cout << vect[nIndex] << " ";

    cout << endl;

}

**Q25.** Change the member function ResetValue() to make it impossible for the function to be able to alter the value of any of its own class data members; write the revised function in the box.

**void ResetValue() const { m\_nValue = 0; }**

class A

{

public:

    float Value1, Value2, Value3;

    void ResetValue(){ m\_nValue = 0; }

};

Also, how can you derive class B from class A in a way that makes ResetValue() inaccessible from objects of class B?

Answer: this can be done by using

**Protected Inheritance (or Private)**

**// end of questions**