91.201 Computing III (Fall 2015) Instructor: Anna Rumshisky

**Exercise #4 – Inheritance and Subtyping. Virtual Functions and Dynamic Binding.**

|  |  |
| --- | --- |
| Handed out: 10/8/15 | Due: 10/13/15 |

Questions

Consider the following class definitions:

class Base { private: int base\_data; public:

Base(int data = 512) : base\_data(data) {

}

int getData() {

return base\_data;

}

virtual void print() { cout << "Base print() called" << endl; }

void baseMethod() { cout << "baseMethod called" << endl; }

};

class Derived : public Base { private:

int derived\_data; public:

Derived(int data = 1024) : derived\_data(data) {

}

int getData() {

return derived\_data;

}

virtual void print() { cout << "Derived print() called" << endl; }

void derivedMethod() { cout << "derivedMethod called" << endl; }

};

91.201 Computing III (Fall 2015) Instructor: Anna Rumshisky

[24 pts] For each of the following statements, please indicate whether (1) it would compile or produce a compilation error, and (2) if it compiles, what function would be invoked:

1. Base b;

This would compile and would call the default constructor for the Base class.

1. Derived d;

This would compile and would call the default constructor for the Base class and then would call the default constructor.

1. Base \*p = &d;

This will compile, but it won’t call any functions.

1. Derived \*q = &b;

This will not compile because you can’t convert from a base to a derived class.

1. cout << b.getData() << endl;

This will call the Base classes’ getData function and will print out 512.

1. cout << d.getData() << endl;

This will call the Derived classes’ getData function and will print out 1024.

1. d.baseMethod();

This will call the Base classes’ baseMethod since d was changed to point to an object of type Base.

1. cout << p­>getData() << endl;

This will compile and will call the base classes getData function.

1. p­>derivedMethod();

This won’t compile because although p points to a class of type Derived, it doesn’t have access to functions that are not part of the base class.

1. b.print();

This will compile and will call the Base classes print function.

1. d.print();

This will compile and will call the Derived classes print function.

1. p­>print();

This will call the derived classes print function, only because it is a virtual function. If it were not a virtual function it would call the Base class version if it existed.