Inheritance and Polymorphism

Section 11.13, 15.1

What is the Output?

```
class Base
{
  public:
    Base() {cout <<"Entering the base.\n";}
    Base(char *str) { cout << "This base is "
                           << str << ".\n"; }
    ~Base() {cout << "Leaving the base.\n";}
};
class Camp : public Base
{
  public:
    Camp() { cout << "Entering the camp.\n";}
    Camp(char *str1, char *str2) : Base(str1)
    { cout << "The camp is " << str2 << ".\n";}
    ~Camp() {cout << "Leaving the camp.\n";}
};
int main()
{
  Camp cOutpost("secure", "secluded");
  return 0;
}
```

Overriding Base Class Functions

- A derived class can override a member function of its base class by defining a derived class member function with the same name and parameter list

Example

```
class Person
{
  private:
    string name;
  public:
    Person() { setName(""); }
    Person(string pName) { setName(pName); }
    void setName(string pName) { name = pName; }
    string getName() { return name; }
};
```

```
class Faculty : public Person
{
  private:
    Discipline department;
  public:
    Faculty(string fname, Discipline d)
      {setName(fname); setDepartment(d); }
    void setDepartment(Discipline d)
      { department = d; }
    Discipline getDepartment()
      { return department; }
};
```

```
class TFaculty : public Faculty
{
  private:
    string title;
  public:
    TFaculty(string fname, Discipline d, string title)
      : Faculty(fname, d)
    {
      setTitle(title);
    }
  void setTitle(string title) { this->title = title; }
   string getName() { return title + " " +
                      Person::getName(); }
};
```

What is the Output

```
int main ()
{
  TFaculty cTFaculty(”Khoja", COMPUTER_SCIENCE, ”DR.");
  cout << cTFaculty.getName() << endl;


  Faculty *pAdvisor = new Faculty(“Williams",
  COMPUTER_SCIENCE);
  cout << pAdvisor->getName() << endl;
  return 0;
}
```

- List all of the functions that are called. Include the class name.

Type Compatibility

- Objects of a derived class can be used wherever objects of a base class object are expected
- Rules for pointers and objects:
  - A derived class pointer can always be assigned to a base class pointer
  - A type cast is required to perform the opposite assignment
    - This could cause an ERROR!!!

Example

```
class Base
{
  public:
    int i;
    Base(int k) {i = k;}
};
class Derived : public Base
{
  public:
    double d;
    Derived(int k, double g) : Base(k) { d = g;}
};
```

Which are allowed?

- Base *pb = new Base(5);
- Derived *pd = new Derived(6, 10.5);
- Base *pb1 = pd;
- Base *pb2 = new Derived(7, 11.5);
- Derived *pd1 = static_cast<Derived *>(pb1);
- cout << pd1->d;
- pd = static_cast<Derived *>(pb);
- cout << pd->d;

What is the Output?

```
class Base
{
  protected:
    int baseVar;
  public:
    Base(int val = 2) { baseVar = val; }
    int getVar() { return baseVar; }
};
class Derived : public Base
{
  private:
    int deriVar;
  public:
    Derived(int val = 100) { deriVar = val; }
    int getVar() { return deriVar; }
};
int main()
{
  Base *pObject;
  Derived object;
  pObject = & object;
  cout << pObject->getVar() << endl;
  return 0;
}
```