Lab 07
Polymorphism

Section 1: Guess program outputs.

1. What will the following program display?

```
#include <iostream>
#include <memory>
using namespace std;

class First
{
protected:
  int a;
public:
  First (int x = 1) { a = x; }
  virtual void twist() { a *= 2; }
  int getVal() { twist(); return a; }
};

class Second : public First
{
private:
  int b;
public:
  Second(int y = 5) { b = y; }
  virtual void twist() { b *= 10; }
};

int main()
{
  shared_ptr<First> object1 = make_shared<First>();
  shared_ptr<Second> object2 = make_shared<Second>();
  cout << object1->getVal() << endl;
  cout << object2->getVal() << endl;

  return 0;
}
```

2. What will the following program display?

```
#include <iostream>
#include <memory>
using namespace std;

class Base
{
protected:
  int baseVar;
public:
  Base(int val = 2) { baseVar = val; }
  int getVar() const { return baseVar; }
};

class Derived : public Base
{
private:
```

```
  int deriVar;
public:
  Derived(int val = 100) { deriVar = val; }
  int getVar() const { return deriVar; }
};

int main()
{
    shared_ptr<Base> optr = make_shared<Derived>();
    cout << optr->getVar() << endl;

    return 0;
}
```

Section 2: Review Questions and Exercises

1. Explain the difference between static binding and dynamic binding.

2. Are virtual functions statically bound or dynamically bound?

3. How can you tell from looking at a class declaration that a virtual member function is pure?

4. What makes an abstract class different from other classes?

Section 3: Programming Challenges

1. Define two derived classes which are IncrSorter and DecrSorter to complete the program below that sort arrays in ascending order and descending order, respectively.
The Sorter class that has an array of integers as a member variable, a pure virtual member function
bool compare (int x, int y) = 0;
that compares its two parameters and returns a boolean value, and a member function
void sort()
that uses the comparison defined by the compare virtual function to sort the array.
The sort() function will swap a pair of array elements a[k] and a[j] if
compare (a[k], a[j])
returns true.

```
#include <algorithm>
#include <iostream>
using namespace std;

class Sorter
{
private:
    int *p;
    int size;
    void sort(int n);
    virtual bool compare(int x, int y) = 0;
public:
    void setArray(int *arr, int size)
    {
        p = arr;
        this->size = size;
    }
    void sort()
    {
        sort(size);
    }
};

void Sorter::sort(int n)
{
    if (n <= 1)
        return;
    //find pos of largest in p[0..n-1]
    //and put it at end of p[0..n-1]
    int maxPos = 0;
    for (int k = 1; k < n; k++)
        if (compare(p[k], p[maxPos]))
        {
            maxPos = k;
        }
    swap(p[n-1], p[maxPos]);
    sort(n-1);
}

void printArray(int [], int);

// Example of usage
int main( )
{
    int a[5] = {23, 56, 23, -45, 52};
```

```
    IncrSorter incSorter;
    incSorter.setArray(a, 5);
    incSorter.sort();
    printArray(a, 5);

    DecrSorter decSorter;
    decSorter.setArray(a, 5);
    decSorter.sort();
    printArray(a, 5);

    return 0;
}

void printArray(int a[ ], int size)
{
    for (int k = 0; k < size; k++)
        cout << a[k] << "  ";
    cout << endl;
}
```

2. Find all errors in the following fragment of code.

```
class MyClass
{
public:
  virtual void myFun() = 0;
  { cout << "Hello"; }
};
```