

ECE 462 Written Assignment 1

Due: 12:20PM, September 22, 2008

Name:

PUID:

1 C++ Polymorphism (Outcome 3)

What is the output of this program?

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

class Person
{
private:
    string p_name;
public:
    Person(string n) : p_name(n)
    { }
    void print()
    {
        cout << "name: " << p_name << endl;
    }
};

class Student: public Person
{
private:
    string s_school;
public:
    Student(string n, string sch) : Person(n), s_school(sch)
    { }
    void print()
    {
        Person::print();
        cout << "school: " << s_school << endl;
    }
};

int main(int argc, char * argv[])
```

```

{
    Person * per[3];
    per[0] = new Person("John");
    per[1] = new Student("Amy", "Purdue");
    per[2] = new Person("Tom");
    Student * stu[2];
    stu[0] = new Student("Jennifer", "Indiana");
    stu[1] = new Student("Charlie", "Lafayette");
    per[1] = stu[1];
    stu[0] = per[0];
    stu[0] -> print();
    return 0;
}

```

2 Java Derived Class (Outcome 3)

What is the output of this program?

```

import java.io.*;
import java.util.*;

class Person
{
    private String p_name;
    protected int p_val;
    public Person(String n) { p_val = 0; p_name = n; }
    public void func1() { p_val ++; }
    public void func2() { p_val += 2; }
    public int getVal() { return p_val; }
}

class Student extends Person
{
    private String s_school;
    public Student(String n, String sch)
    {
        super(n);
        s_school = sch;
    }
    public void func1() { p_val += 3; }
}

class CollegeStudent extends Student
{
    private String cs_major;
    public CollegeStudent(String n, String sch, String ma)
    {
        super(n, sch);
        cs_major = ma;
    }
}

```

```

    }
    public void func2() { p_val += 4; }
};

class outcome342
{
    public static void main(String[] args)
    {
        Person [] per = new Person[3];
        per[0] = new Person("John");
        per[1] = new Student("Amy", "Purdue");
        per[2] = new CollegeStudent("Tom", "West Lafayette", "ECE");
        Student [] stu = new Student[2];
        stu[0] = new Student("Jennifer", "Indiana");
        stu[1] = new CollegeStudent("Mary", "Indiana", "Math");
        per[1].func1();
        per[1].func2();
        stu[1].func1();
        stu[1].func2();
        System.out.println(per[1].getVal());
        System.out.println(stu[1].getVal());
    }
}

```

3 Inheritance and Class (Outcome 3)

Which statement is correct?

- A. If a C++ class is abstract, all methods must be purely virtual.
- B. In Java, a private attribute can be seen by derived classes; a protected attribute cannot be seen by derived classes.
- C. Polymorphism means classes can have derived classes.
- D. The only purpose of creating a class is to encapsulate data.
- E. A derived class has all the public methods from the base class.

4 C++ Abstract Class (Outcome 3)

How to make Shape an abstract class and Triangle must override the getArea method?

```

#include <iostream>
#include <string>
using namespace std;

class Shape {
public:
    Shape() { }
    virtual ~Shape() { }
    // >>>>>
    // getArea must be overridden in derived classes
    // <<<<<<
};

class Triangle : public Shape
{
public:
    Triangle(double h, double w)
    {
        width = w;
        height = h;
    }
    virtual double getArea() {
        return (0.5 * height * width);
    }
    virtual ~Triangle() { }
private:
    double height;
    double width;
};

int main(int argc, char * argv[])
{
    Shape * sobj1 = new Triangle(4.0, 5.0);
    double a = sobj1 -> getArea();
    cout << a << endl;
    return 0;
}

```

5 Java Classes (Outcome 3)

What is the output of this program?

```

import java.io.*;
import java.util.*;

class Person
{
    private String p_name;
    public Person(String n) { p_name = n; }
}

```

```

    public void print()
    {
        System.out.println("name: " + p_name);
    }
}

class Student extends Person
{
    private String s_school;
    public Student(String n, String sch)
    {
        super(n);
        s_school = sch;
    }
    public void print()
    {
        super.print();
        System.out.println("school: " + s_school);
    }
}

class outcome362
{
    public static void main(String[] args)
    {
        final int numPerson = 3;
        Person [] per = new Person[numPerson];
        per[0] = new Person("John");
        per[1] = new Student("Amy", "Purdue");
        per[2] = new Student("Tom", "West Lafayette");
        for (int scnt = 0; scnt < numPerson; scnt ++)
        {
            per[scnt].print();
        }
    }
}

```

6 C++ Constructor and Destructor (Outcome 3)

What is the output of this program?

```

#include <iostream>
#include <string>
using namespace std;

class Person
{
protected:
    static int counter;

```

```

public:
    Person()
    { counter ++; }
    virtual ~Person()
    { counter --; }
    int getCounter()
    { return counter; }
};

int Person::counter = 0;

class Student: public Person
{
public:
    Student()
    { counter ++; }
};

int main(int argc, char * argv[])
{
    const int numPerson = 4;
    Person * per[numPerson];
    per[0] = new Person;
    per[1] = new Student;
    per[2] = new Student;
    per[3] = new Person;
    {
        Person p1;
        Student s1;
    }
    delete per[1];
    delete per[2];
    cout << per[0] -> getCounter() << endl;
    return 0;
}

```

7 C++ Polymorphism (Outcome 3)

What word (or words) should be added in front of a C++ member function to enable polymorphism?

8 Inheritance, Interface, and Implementation (Outcome 3)

Which statement is correct?

- A. A class provides an interface through which messages can be sent.

- B. If a Java class has a virtual function, this class is abstract.
- C. An abstract class cannot have any constructor.
- D. If an object X is not declared as an attribute inside a class Y, this object X is not allowed to send any message to any object of class Y.
- E. If a class has no virtual function, this class cannot have derived classes.

9 C++ Template (Outcome 4)

Write the declaration of the class, the private attributes, and the constructor.

```
#include <iostream>
#include <string>
#include <stack>
using namespace std;

class Student
{
public:
    int s_ID;
    string s_name;
    Student(int id, string name): s_ID(id), s_name(name)
    { }
    void print()
    {
        cout << "ID: " << s_ID << " name: " << s_name << endl;
    }
};

// >>>>
// write the class declaration
// <<<<<
{
private:
    // >>>>
    // declare the attributes
    // <<<<<
public:
    // >>>>
    // define the constructor
    // <<<<<
    void insert(ECE462List<TPL> * node)
    {
        el_next = node;
    }
    void print()
    {
        el_content.print();
        ECE462List * node = el_next;
    }
};
```

```

        while (node != 0)
        {
            (node -> el_content).print();
            node = node -> el_next;
        }
    }
};

int main(int argc, char * argv[])
{
    Student s1(12345, "Amy");
    Student s2(65412, "John");
    Student s3(25789, "Tom");
    Student s4(78941, "Jennifer");
    ECE462List<Student> * head = new ECE462List<Student>(s1);
    ECE462List<Student> * n2 = new ECE462List<Student>(s2);
    ECE462List<Student> * n3 = new ECE462List<Student>(s3);
    ECE462List<Student> * n4 = new ECE462List<Student>(s4);
    head -> insert(n2);
    n2 -> insert(n3);
    n3 -> insert(n4);
    head -> print();
    delete head;
    delete n2;
    delete n3;
    delete n4;
    return 0;
}

```

10 C++ Stack (Outcome 4)

What is the output of this program?

```

#include <iostream>
#include <string>
#include <stack>
using namespace std;

class Student
{
public:
    int s_ID;
    string s_name;
    Student(int id, string name): s_ID(id), s_name(name) { }
};

int main(int argc, char * argv[])
{
    stack<Student> ecestudent;
}

```



```

Student s1(12345, "Amy");
Student s2(65412, "John");
ecestudent.push(s1);
ecestudent.push(s2);
s1.s_ID = 98741;
s2.s_name = "Tom";
cout << ecestudent.top().s_name << endl;
return 0;
}

```

11 Container Classes (Outcome 4)

Which statement is correct?

- A. In a map, the values and the keys must be unique. Thus, it must be a one-to-one mapping.
- B. Once a vector object is created, the first element cannot be deleted.
- C. In a C++ list, duplicate elements are not allowed. The elements must be unique.
- D. Elements can be inserted or removed only at the beginning of a list.
- E. In a Java list, duplicate elements are allowed.

12 Java Set (Outcome 4)

What is the output of this program?

```

import java.io.*;
import java.util.*;

class Student
{
    public String s_name;
    public int s_ID;
    public Student(String name, int id)
    {
        s_name = name;
        s_ID = id;
    }
    public String toString()
    {
        String str = s_name + " " + String.valueOf(s_ID);
        return str;
    }
}

```

```

class outcome471
{
    public static void main(String[] args)
    {
        Set<Student> ecestudent = new HashSet<Student>();
        Student s1 = new Student("Amy", 1);
        Student s2 = new Student("John", 2);
        Student s3 = new Student("Amy", 3);
        Student s4 = new Student("John", 4);
        Student s5 = new Student("Jennifer", 5 );
        ecestudent.add(s1);
        ecestudent.add(s2);
        ecestudent.add(s3);
        ecestudent.add(s4);
        ecestudent.add(s3);
        ecestudent.add(s4);
        ecestudent.add(s5);
        System.out.println(ecestudent);
    }
}

```

13 C++ Queue (Outcome 4)

Implement a C++ queue using list. For simplicity, this queue can handle only strings.

```

#include <iostream>
#include <string>
#include <list>
using namespace std;

class ECE462Queue
{
private:
    list<string> eq_list;
public:
    ECE462Queue()
    {
    }
    void addEnd(string str)
    {
        // >>>>>
        // add the string to the end of the queue
        // <<<<<<
    }
    string removeFront()
    {
        // >>>>>
        // remove the string at the beginning of the queue
        // <<<<<<
    }
}

```

```

        // you can add code but cannot remove any existing code
        // return an empty string if the queue is empty
        return string();
    }
};

int main(int argc, char * argv[])
{
    ECE462Queue que;
    que.addEnd("this");
    que.addEnd("is");
    que.addEnd("a");
    que.addEnd("nice");
    que.addEnd("day");
    cout << que.removeFront() << endl;
    cout << que.removeFront() << endl;
    cout << que.removeFront() << endl;
    cout << que.removeFront() << endl;
    cout << que.removeFront() << endl;
    cout << que.removeFront() << endl;
    cout << que.removeFront() << endl;
    cout << que.removeFront() << endl;
    cout << que.removeFront() << endl;
    return 0;
}

```

14 Java List (Outcome 4)

What is the output of this program?

```

import java.io.*;
import java.util.*;
class outcome442
{
    public static void print(List<String> lis)
    {
        System.out.println("list size is: " + lis.size());
        Iterator p = lis.iterator();
        while (p.hasNext())
        {
            System.out.println(p.next());
        }
    }
    public static void main(String[] args)
    {
        List<String> ece462 = new ArrayList<String>();
        ece462.add("c++");
        ece462.add("fall");
        ece462.add("2008");
    }
}

```

```

        ece462.add("java");
        print(ece462);
    }
}

```

15 C++ Vector (Outcome 4)

Please declare the iterator in function print.

```

#include <iostream>
#include <vector>
using namespace std;
void print(vector<string> & v)
{
    cout << "vector size is: " << v.size() << endl;
    // >>>>>
    // declare and define iterator p
    // <<<<<<
    while ( p != v.end() )
    {
        cout << (*p) << endl;
        p ++;
    }
}
int main(int argc, char * argv[])
{
    vector<string> vec;
    vec.push_back("hello");
    vec.push_back("ece462");
    vec.push_back("C++");
    vec.push_back("Java");
    print(vec);
    return 0;
}

```

16 Container Classes (Outcome 4)

Which statement is correct?

- A. An element can be inserted anywhere in a list.
- B. An element can be inserted anywhere in a stack.
- C. Elements can be inserted or removed only at the end of a list, not anywhere else.
- D. Elements can be inserted only at the end of a list and removed at the beginning of a list.
- E. In a map, the values must be unique and the keys can be the same. Thus, it can be a one-to-many mapping.

17 C++ Parameter Passing

What is the output of this program?

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

class Person
{
public:
    string p_name;
    Person(string n): p_name(n) { }
    string getName() { return p_name; }
};

class Student: public Person
{
public:
    string s_school;
    Student(string n, string s): Person(n), s_school(s) { }
    string getSchool() { return s_school; }
};

void changeNames(Person & p1, Person & p2)
{
    p1.p_name = p2.p_name;
}

int main(int argc, char * argv[])
{
    const int numPerson = 4;
    Person * per[numPerson];
    per[0] = new Person("John");
    per[1] = new Student("Amy", "Purdue");
    per[2] = new Student("Tom", "West Lafayette");
    per[3] = new Person("Emily");
    changeNames((*per[0]), (*per[1]));
    changeNames((*per[2]), (*per[0]));
    changeNames((*per[3]), (*per[2]));
    cout << per[3] -> getName() << endl;
    return 0;
}
```

18 C++ Copy Constructor

Write the copy constructor to perform deep copy.

```

#include <iostream>
using namespace std;
class X {
    int* x_data;
    int x_size;
public:
    X(int* dat, int sz) : x_size(sz) {
        x_data = new int[sz];
        for (int i = 0; i < sz; i++)
            { x_data[i] = dat[i]; }
    }
    // >>>>>
    // write the copy constructor
    // <<<<<<
    X & operator = (const X& xobj) {
        if (this != & xobj) {
            delete [] x_data;
            x_size = xobj.x_size;
            x_data = new int [x_size];
            for (int i = 0; i < x_size; i++)
                { x_data[i] = xobj.x_data[i]; }
        }
        return *this;
    }
    virtual ~X() { delete [] x_data; }
};
int main()
{
    int freshData[5] = {1, 2, 3, 4, 5};
    X x1(freshData, 5);
    X x2 = x1;
    X x3(freshData, 5);
    x3 = x2;
    x2 = x2;
    return 0;
}

```

19 Overload Resolution in Java

What is the output of this program?

```

class Person
{
    private String p_name;
    public Person(String n) { p_name = n; }
    public void print()
    {
        System.out.println("name: " + p_name);
    }
}

```

```

    }
}

class Student extends Person
{
    private String s_school;
    public Student(String n, String sch)
    {
        super(n);
        s_school = sch;
    }
    public void print()
    {
        super.print();
        System.out.println("school: " + s_school);
    }
}

class outcome031
{
    static void func1(Person p1, Person p2) {
        System.out.println("first func1");
    }
    static void func1(Person p, Student s) {
        System.out.println("second func1");
    }
    static void func1(Student s, Person p) {
        System.out.println("third func1");
    }
    public static void main(String[] args)
    {
        Student stu1 = new Student("John", "Purdue");
        Student stu2 = new Student("Amy", "West Lafayette");
        func1(stu1, stu2);
    }
}

```

20 Java Array

Fill in the code to create an array of 5 Student objects so that the students' IDs are 0 to 4.

```

import java.io.*;
import java.util.*;

class Student
{
    public int s_ID;
    public Student(int id)
    {

```

```

        s_ID = id;
    }
    public String toString()
    {
        String str = String.valueOf(s_ID);
        return str;
    }
}

class outcome041
{
    public static void main(String[] args)
    {
        int numStudents = 5;
        // >>>>>
        // create five Student objects
        // <<<<<
        for (int scnt = 0; scnt < numStudents; scnt ++)
            {
                System.out.println(stus[scnt]);
            }
        // output:
        /*
        0
        1
        2
        3
        4
        */
    }
}

```