Week 3 Exercises Submit

Sandra Batista

1.1-1.2

1. Exercise: grocerylist.cpp

1. Write a **copy constructor** for GroceryList
2. Write a **copy assignment operator** for Grocery List

Extra practice

1. Write an '==' operator for GroceryList
2. Write [] operator for GroceryList. Make sure to include const and non-const versions. (why
3. Write + operator for GroceryList
4. Write += operator for GroceryList

Code: https://github.com/sandraleeusc/csci104_fall2020_lecture/

```
// for main inside grocerylist.cpp
//once you have written appropriate
//functions, you can change main to this

int main() {
  GroceryList list1, list2;
  list1.addItem("apples");
  list1.addItem("bananas");
  list1.addItem("peaches");
  list1.printList();
  list2.addItem("onions");
  list2.addItem("peppers");
  list2.addItem("broccoli");
  GroceryList list3 = list1;
  cout << boolalpha << (list1 == list3) << endl;
 GroceryList list4 = list1 + list2;
  list4.printList();
  cout << list4[3] << endl;
  list4[3] = "oatmeal";
  list4.printList();
}
```

2. T/F and Multiple Choice Inheritance questions

Submit your solutions to these previous exam T/F and multiple choice questions on inheritance: https://github.com/sandraleeusc/csci104_fall2020_lecture/blob/master/Inheritance_q.pdf

(1) [12 points]
True or False Questions. 2 points each - full credit or no credit.
Suppose we have the following inheritance specified in a library file.

```
class A { /* public, protected, and private data and functions*/}
class B : protected A { /* public, protected, and private data and functions*/}
class C : private B { /* public, protected, and private data and functions*/}
```

(a) False A base class is type-compatible with a derived class.
(b) False A derived class is type-compatible with a base class when protected inheritance is used.
(c) True When argument objects are passed to a function by value, a copy constructor is called.
(d) False When public inheritance is used, only public functions of the derived class, but none from the base class are available to client classes.
(e) True Class C may access all the protected and public data and functions of Class A and Class B.
(f) True All clients of class B can only access the public data and functions of class B only and none of class A.

(2) [12 points]
Multiple Choice Questions. 2 points each - full credit or no credit

```
// class SampleClass defined in SampleClass.h
SampleClass A; /*1*/
SampleClass B = A; /*2*/
B = A; /*3*/
```

(a) In the line above labeled /*2*/ what function is called? Select one.
  (a) Copy constructor
  (b) Assignment operator
  (c) Default constructor
(b) In the line above labeled /*3*/ what function is called? Select one.
  (a) Copy constructor
  (b) Assignment operator
  (c) Default constructor
(c) Which of the following prototypes demonstrates best practices for writing a copy constructor for the ClassA? Choose the best answer
  (a) ClassA(ClassA& copy);
  (b) ClassA(const ClassA& copy);
  (c) ClassA(const ClassA copy);
(d) ClassB dynamically allocates an array of integers in its constructor. In addition to a default constructor, what are the smallest set of functions that ClassB should implement to avoid memory leaks?
  (a) ClassB should implement a destructor.
  (b) ClassB should implement an assignment operator and destructor.
  (c) ClassB should implement a copy constructor, assignment operator, and destructor.
(e) In the class LinkedList a student has placed the following prototype for overloading the insertion operator:
  `friend ostream& operator<< (ostream& o, const LinkedList &ll);`
  Select all statements that are true about this prototype
  (a) The function will not be able to access private data and functions of the Linked List class.
  (b) The function may change the LinkedList passed as parameter to print.
  (c) The ostream may be modified for the insertion.
  (d) Chaining insertion calls will be possible. (Chaining example: cout << list1 << list2;)
(f) Private data of a base class may be initialized in the following ways: (Circle all that apply)
  (a) By calling the constructor of the base class within the constructor of the derived class
  (b) By calling the constructor of the base class within the initialization list of the constructor of the derived class
  (c) By setting the data directly within the constructor of the derived class since the derived class has a copy of it also.

3. Virtual Function Exercises

(a)
```
class Packager {
public:
  virtual ~Packager() {}
  string package(string& s) {
    return material() + s + material();
  }
protected:
  virtual string material() { return "-"; }
private:
};
class APackager : public Packager {
protected:
  string material() { return "A"; }
};
class DblPackager : public APackager {
public:
  string package(string& s) {
    return material() + Packager::package(s) + material();
  }
};
```

```
int main()
{
  Packager *p = new Packager;
  APackager *a = new DblPackager;
  DblPackager *d = new DblPackager;
  string s1 = "123";
  cout << p->package(s1) << endl;
  cout << a->package(s1) << endl;
  cout << d->package(s1) << endl;
  delete p; delete a; delete d;
  return 0;
}
```

What is output by main() and why?

(b) https://bytes.usc.edu/cs104/resources/midterm-a.pdf Question 4 on page 7

(4) [20 points]
Here is a piece of code. Tell us what it outputs. (You will get partial credit for partially correct answers.)

```
class Question {
public:
  Question(int v) : val(v) { }
  virtual ~Question() { cout << "d1" << endl; }
  virtual string studentResponse() = 0;
  int getValue() { return val; }

private:
  int val;
};

class NonTrivialQuestion : public Question {
public:
  NonTrivialQuestion() : Question(10) { }
  NonTrivialQuestion(int v) : Question(v) { }
  ~NonTrivialQuestion() { cout << "d2" << endl; }
  string studentResponse() { return "I got this!"; }
  int getValue() { return 15 + Question::getValue(); }
};

class DifficultQuestion : public NonTrivialQuestion {
public:
  DifficultQuestion() : NonTrivialQuestion() { }
  ~DifficultQuestion() { cout << "d3" << endl; }
  string studentResponse()
      { return "When are office hours?"; }
};

int main()
{
  Question* p[2];
  p[0] = new NonTrivialQuestion(15);
  p[1] = new DifficultQuestion;
  for(int i=0; i < 2; i++){
    cout << p[i]->getValue() << endl;
    cout << p[i]->studentResponse() << endl;
  }
  NonTrivialQuestion* q[2];
  q[0] = new NonTrivialQuestion(15);
  q[1] = new DifficultQuestion;
  for(int i=0; i < 2; i++){
    cout << q[i]->getValue() << endl;
    cout << q[i]->studentResponse() << endl;
  }
  delete p[1];
  return 0;
}
```

Output:

15
I got this
10
When are office hours?

30
I got this
25
When are office hours?
d3
d2
d1

NonTrivialQuestion q5;
Question::getValue
p[0]=&q5;
polymorphism
NonTrivalQuestion::getValue
// delete q[1];
// After main exit destructor for q5 call

The code for tracing is available here:
https://github.com/sandraleeusc/csci104_fall2020_lecture

Trace the output of functiontrace.cpp

- The output is in function_trace_output
- You need to understand what function is being called on each line and why.
- You should understand what function printed each statement. Other functions are called that do no print anything.
- You can add print statements to standard error, cerr
- To compile: g++ --std=c++17 -o test functiontrace.cpp
- To run and redirect standard error to a file:

./test 2> testing_outputfile