University of Illinois at Urbana-Champaign
Department of Computer Science

First Examination

CS 225 Data Structures and Software Principles
Spring 2014
Tuesday, February 25, 7-10p

Name:
NetID:
Lab Section (Day/Time):

- This is a **closed book** and **closed notes** exam. No electronic aids are allowed.
- You should have 4 problems total on 16 pages. The last sheet is scratch paper; you may detach it while taking the exam, but must turn it in with the exam when you leave.
- The points assigned to each problem are a *rough* estimate of the time it should take you to solve the problem. Use your time wisely.
- Unless otherwise stated in a problem, assume the best possible design of a particular implementation is being used.
- Unless the problem specifically says otherwise, (1) assume the code compiles, and thus any compiler error is an exam typo (though hopefully there are not any typos), and (2) assume you are NOT allowed to write any helper methods to help solve the problem, nor are you allowed to use additional arrays, lists, or other collection data structures unless we have said you can.
- We will be grading your code by first reading your comments to see if your plan is good, and then reading the code to make sure it does exactly what the comments promise.
- Please put your name at the top of each page.

Problem	Points	Score	Grader
1	30		
2	20		
3	30		
4	20		
Total	100		

1. **[Pointers, Parameters, and Miscellany – 30 points].**

MC1 (2.5pts)

What common bug is shown in the following code snippet?

```
int * foo;
*foo = 5;
```

(a) syntax error
(b) type mismatch
(c) uninitialized pointer
(d) dereferenced NULL
(e) None of the above phrases describes the bug.

MC2 (2.5pts)

What is the best interpretation of the following Valgrind output?

```
Address 0x4c78598 is 8 bytes after a block of size 32 alloc'd
```

(a) We tried to delete an object from memory that was on the stack.
(b) We tried to use an uninitialized value.
(c) We tried to delete a heap object twice.
(d) We tried to dereference NULL.
(e) We tried to access memory outside the bounds of a dynamic array.

MC3 (2.5pts)

Suppose a class called `Sprite` has implemented a destructor. Which of the following statements is also true of a well-designed `Sprite` class?

(a) An object of type `Sprite` should not be passed by value.
(b) The `Sprite` class will provide an overloaded `operator!=` for use in its `operator=` function.
(c) The `Sprite` class will contain at least 2 constructors.
(d) Exactly 2 of items (a), (b), and (c) are true.
(e) None of the above statements are true.

MC4 (2.5pts)

Which answer is true about this overloaded assignment operator for the `Rectangle` class?

```
1 Rectangle & operator=(const Rectangle & rect) {
2         if (this != &rect)  {
3                 clear();
4                 copy(rect);
5         }
6         return *this;
7 }
```

(a) The code will not compile

(b) Line 1 is incorrect. It should be changed to:
```
1 void operator=(const Rectangle & rect) {
```

(c) Line 1 and 6 are incorrect. They should be changed to:
```
1 Rectangle * operator=(const Rectangle & rect) {
6 return this;
```

(d) Line 1 is incorrect. It should be changed to:
```
1 Rectangle operator=(const Rectangle rect) {
```

(e) This code is a reasonable definition of `Rectangle`'s overloaded assignment operator.

MC5 (2.5pts)

Consider the following code and assume that the standard `iostream` library has been included:

```
void stir(int & x, int * y) {
  x = 7;
  y = new int(10);
}

int main() {
  int w;     int * v = NULL;
  stir(w, v);

  cout << w << ", " << *v << endl;
  delete v;   return 0;
}
```

What is the result of compiling and executing these statements?

(a) `7, 10` is sent to standard out.

(b) This code results in a compile error.

(c) This code results in a runtime error.

(d) This code compiles and runs, but it has undefined behavior.

(e) None of these options is correct.

MC6 (2.5pts)

Which of the following statements about a class's destructor is FALSE?

(a) We never call the destructor, but rather, we provide it for the system to use.

(b) A destructor is only invoked by the system when an instance of a class is deleted (using the delete keyword).

(c) Destructors typically contain a sequence of delete statements.

(d) The role of a destructor is to free dynamically allocated memory.

(e) All of items (a) through (d) are TRUE.

MC7 (2.5pts)

```
template <class E, class F>
bool comp(E a, F b) {
    return (a > b);
}
```

Which of the following statements will give a compiler error?

(a) `bool b = comp<int, string>(100, "my 225 exam grade");`

(b) `bool b = comp<int, int>(4, 8);`

(c) `bool b = comp<int, double>(4, 4.1);`

(d) Exactly two of these answers fail to compile.

(e) All three of these answers compile.

MC8 (2.5pts)

Suppose we are implementing a `List` class using a singly linked list with head and tail pointers. Which of the following operations can be implemented in $O(1)$ time?

I Insert item at the front of the list
II Insert item at the rear of the list
III Delete front item from list
IV Delete rear item from list

(a) I and II

(b) I and III

(c) III and IV

(d) I, II, and III

(e) All operations can be performed in constant time.

MC9 (2.5pts)

Consider the following class definitions:

```
class Season{
public:
        virtual void adjustTemp(int change);
private:
        int temp;
};
class Winter: public Season {
public:
        void makeColder(int change);
};
```

Where could the assignment `temp += change;` appear for the private variable `temp`?

(a) Both `adjustTemp` and `makeColder` can make the assignment.

(b) `adjustTemp` can make the assignment, but `makeColder` cannot.

(c) `makeColder` can make the assignment, but `adjustTemp` cannot.

(d) Neither `makeColder` nor `adjustTemp` can make the assignment.

(e) The answer to this question cannot be determined from the given code.

MC10 (2.5pts)

Which of the following is a correct function signature for the overloaded addition operator for the sphere class, if we want that operator to return a sphere whose radius is the sum of the radii of the object and its parameter?

(a) `sphere sphere::operator+(const sphere & right) const;`

(b) `sphere & sphere::operator+();`

(c) `sphere sphere::operator+(const sphere & left, const sphere & right);`

(d) More than one of (a), (b), (c), could be used.

(e) None of (a), (b), (c), are appropriate.

MC11 (2.5pts)

Suppose class `pictureRep` contains exactly one pure virtual function: the overloaded parentheses operator, `int operator()(int i, int j)`. Also suppose that class `hardPNG` is a public `pictureRep` that implements `operator()`.

Which of the following C++ statements will certainly result in a compiler error?

(a) `hardPNG * a = new pictureRep;`

(b) `hardPNG * a = new hardPNG;`

(c)
```
pictureRep * a = new hardPNG;
hardPNG * b;
a = b;
```

(d) Exactly two of these will result in a compiler error.

(e) None of these will result in a compiler error.

MC12 (2.5pts)

Which of the following concepts describe the behavior of the virtual function `speak()` in the following code snippet, if the result is "ruff, moo, oink?"

```
animal ** farm;

farm = new animal*[3];
farm[0] = new dog;
farm[1] = new cow;
farm[2] = new pig;

for (int i=0; i<3;i++)
    farm[i]->speak();
```

(a) `speak()` is encapsulated

(b) `speak()` is polymorphic

(c) `speak()` is templatized

(d) `speak()` is a pure virtual function

(e) None of these options is correct.

2. **[MP2ish – 20 points].**

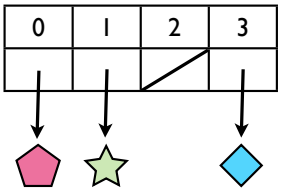
Consider the following partial class definition:

```
class photoCollection
{
  private:
    PNG ** album;
    int albumSize;
    // some helper functions, maybe

  public:
    // constructors and destructor, if necessary

    void replicate(int k);  // for you to implement.  described below.

    // lots of public member functions, including addPic, removePic, etc.
};
```

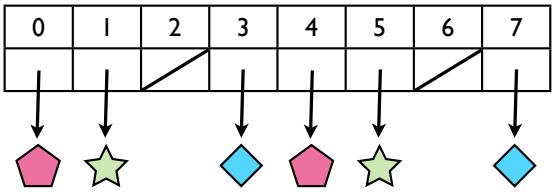
The `album` structure is a dynamically allocated array of `PNG` pointers. `albumSize` is the capacity of the structure, and it will be greater than 0. Every entry in the `album` array should either be `NULL` or it should point to a valid `PNG`.

In this question you will implement a public member function called `replicate` that takes an integer `k` as input, and transforms the album into a new array whose entries are `k` copies of the original, as illustrated here:

If `photoCollection phoCo` contains this data:



Then after `phoCo.replicate(2), phoCo` contains this data:



You will implement `replicate` on the following pages. To grade this problem, we will first read your comments to make sure you intend to do the right thing, and then we'll check your code to make sure it does what your comments say it should. As a result, be sure your comments are coherent, useful, and reflective of your approach to the problem.

(a) (10 points) Describe your plan for implementing `replicate` below. Each step should be a simple sentence using words like "allocate," "copy," and "declare."

i.

ii.

iii.

iv.

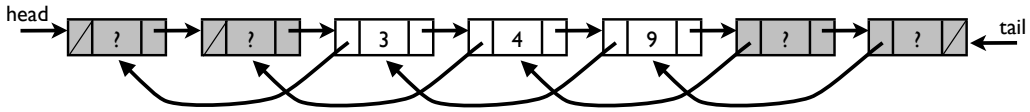
v.

vi.

(b) (10 points) Write the code that implements your plan from part (a).

3. **[Loopy Lists – 30 points].**

In this problem we will implement some of the member functions for a class called `loopy`, illustrated below. `loopy` is a linked memory structure composed of nodes, each of which consists of an integer data element together with two pointers to other nodes. Our implementation of a `loopy` has 4 sentinels, denoted by the grey nodes containing "?" in the illustration. We will maintain the `loopy` so that all data falls between the second and third sentinels. (Recall that we don't care what data is contained in the sentinel nodes.)



This example structure was created by the client code below.

```
loopy L;
L.insertAtFront(9);
L.insertAtFront(4);
L.insertAtFront(3);
```

Each `loopy` node is an instance of the following private structure:

```
struct LNode{
   int data;
   LNode * next;
   LNode * twoPrev;
}
```

(a) (5 points) Write the default (no argument) constructor for the `loopy` class. The only private data members of the class are `LNode * head` and `LNode * tail`. Be sure to comment your code profusely.

(b) (6 points) Help us write public member function `void insertAtFront(int newData)`. This function inserts `newData` into position 1 of the `loopy` (shown above).

```
void loopy::insertAtFront(int newData) {

     // set a pointer to the second sentinel node:

     LNode * prev = _____________;

     // create a new node:

     LNode * newN = _____________;

     // update forward pointers:

     ________________________________________;

     ________________________________________;

     ________________________________________;

     // update twoPrev pointers:

     ________________________________________;

     ________________________________________;

     ________________________________________;
}
```

(c) (3 points) Explain the role of the sentinel nodes in this class. Would your implementation of `insertAtFront` change if they did not exist?

(d) (3 points) Write a member function called `previous` that takes an `LNode * cur` in the `loopy` as input, and returns a pointer to the node immediately before `cur`, so that `(previous(cur)->next == cur)` .

(e) (2 points) Is function `previous` public or private in the `loopy` class?

(f) (5 points) Write a public member function called `printReverse` that prints the data in the `loopy` in reverse order. For the example above, `L.printReverse()` would print the values 9, 4, 3, in that order. You may assume the standard `iostream` has been included.

(g) (2 points) Give an analysis of the running time of the function you wrote in part (f). Briefly explain your response.

(h) (2 points) Does the `loopy` class require a destructor?

(i) (2 points) Does the `LNode` class require a destructor?

4. **[Miscellaneous – 20 points].**

   (a) (6 points) Complete the following (silly) function implementation so that no memory is leaked.

```
void nurture() {
    flower rose;
    flower * pot = &rose;
    flower * hedgerow = new flower[4];
    flower ** plot = new flower * [5];
    plot[0] = &hedgerow[3];
    plot[1] = new flower;
    plot[2] = plot[1];
    plot[3] = pot;
    plot[4] = new flower[8];

    // add code to free each piece of dynamically
    // allocated memory exactly once









}
```

(b) (6 points) Scrutinize the following code. In this problem we are going to ask you to predict and to change its output. (Please assume that the standard `iostream` is included.)

```
class yell{
public:
    void display() {
        cout << "yell" << endl;
    }
};

class w00t:public yell {
public:
    void display() {
        cout << "w00t" << endl;
    }
};

int main() {

    yell * a = new w00t;
    a->display();

    return 0;
}
```

i. (3 points) What is output when the code above is compiled and run?

ii. (3 points) How can you change the class definitions so that the other version of the `display` function is called?

(c) (8 points) For this problem, you will be a code critic. Please answer the questions below about the following `sphere` class member function. `setPictureGetRadius` is supposed to change the member variable `thePicture` and return the current value of member variable `theRadius`. The adapted `sphere` class appears at the end of the exam.

```
double  sphere::setPictureGetRadius(PNG & newPicture) const {

   thePicture = newPicture;
   return theRadius;

}
```

i. (4 points) Comment on the type specification in the parameter list:

- Why is it pass-by-reference?

- Is there a better way to specify that parameter?

ii. (2 points) Comment on the `const` keyword in the function prototype.

iii. (2 points) What does the assignment `thePicture = newPicture;` assume about the `PNG` class? Is this a reasonable assumption?

(d) (0 points) Please give us feedback about the course, entering your responses on items 13 thru 15 of the scantron form you used for your multiple choice responses:

i. On a scale of 1 to 5, how much are you learning in the class? (1 is not much, 5 is a ton)

(a) 1 (b) 2 (c) 3 (d) 4 (e) 5

ii. On a scale of 1 to 5, how is the pace of the course so far? (1 is too slow, 5 is too fast)

(a) 1 (b) 2 (c) 3 (d) 4 (e) 5

iii. On a scale of 1 to 5, rate your general satisfaction with the course. (1 is profoundly dissatisfied, 5 is happy) If your response is not 4 or 5, please suggest a specific improvement we can make.

(a) 1 (b) 2 (c) 3 (d) 4 (e) 5

```
class sphere {
public:
   sphere();
   sphere(double r);

   double getDiameter() const;
   double getRadius() const;
   void setRadius(double r);
   double setPictureGetRadius(PNG & newPicture) const;
private:
   double theRadius;
   PNG thePicture;
};
```

scratch paper