University of Illinois at Urbana-Champaign
Department of Computer Science

First Examination

CS 225 Data Structures and Software Principles
Spring 2009
7p-9p, Tuesday, Feb 24

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

- This is a **closed book** and **closed notes** exam. No electronic aids are allowed, either.
- You should have 5 problems total on 18 pages. The last two sheets are scratch paper; you may detach them while taking the exam, but must turn them in with the exam when you leave.
- 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. In general, complete and accurate comments will be worth approximately 30% of the points on any coding problem.
- Please put your name at the top of each page.

Problem	Points	Score	Grader
1	20		
2	20		
3	17		
4	10		
5	33		
Total	100		

1. **[Pointers, Parameters, and Miscellany – 20 points].**

MC1 (2.5pts)

Consider the following statements:

```
int *p;
int i;
int k;
i = 37;
k = i;
p = &i;
```

After these statements, which of the following will change the value of `i` to 75?

(a) `k = 75;`

(b) `*k = 75;`

(c) `p = 75;`

(d) `*p = 75;`

(e) Two or more of the answers will change `i` to 75.

MC2 (2.5pts)

Consider the following statements:

```
int i = 1;
int k = 2;
int * p1;
int * p2;
p1 = &i;
p2 = &k;
p1 = p2;
*p1 = 3;
*p2 = 4;
cout <<  i  << endl;
```

Which of the following is printed by the output statement (assume `cout` works)?

(a) 1

(b) 2

(c) 3

(d) 4

(e) None of these. The code does not compile.

MC3 (2.5pts)

Consider the following C++ statements:

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

void increment1(int  x) { x = x + 1; }
void increment2(int * x) { *(x) = *x + 1; }
void increment3(int & x) { x = x + 1; }

int main() {
  int x = 1;
  increment1(x);
  increment2(&x);
  increment3(x);
  cout << x << endl;
  return 0;
}
```

What is the printed out when this code is compiled and run?

(a) 1

(b) 2

(c) 3

(d) 4

(e) This code does not compile.

MC4 (2.5pts)

Which of the following is a correct way to declare and initialize a dynamic array of length `max`, each element of which is a `List` whose parameterized type is a `sphere`?

(a) `List<sphere> * myList = new List<sphere>[max];`

(b) `sphere ** myList = new sphere *[max];`

(c) `List<sphere> myList[max];`

(d) More than one of (a), (b), (c), are correct.

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

MC 5 (5 pts)

Find the errors in the following code. Select all that apply, if any.

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

int * myFun(const int & a) {

   int * c = new int(a);
   int b = *c;
   return &b;

}

int main()  {

   int n = 8;

   cout << *(myFun(n)) << endl;

return 0;
}
```

(a) type mismatch

(b) memory leak

(c) invalid return value

(d) NULL pointer dereference

(e) violation of `const`

MC 6 (5pts)

Which of the following functions are typically implemented in an iterator class? Check all that apply, if any.

(a) `operator*`

(b) `operator()`

(c) `operator++`

(d) `operator=` note: an iterator class MIGHT implement this, but I have never seen the need for it. Iterators for most container classes represent indices (integers) or pointers (addresses).

(e) `begin()` note: begin() is a List class member function that returns an iterator.

2. **[The Big Three – 20 points].** Consider the following partial class definition:

```
class picRosters
{
  private:
    BMP ** courses;
    int * enrollments;
    int numCourses;

    // some helper functions

  public:
    // constructors and destructor

    //  operator=   declaration

    // lots of public member functions
};
```

The `courses` structure is a dynamically allocated array of dynamically allocated arrays of `BMP`s. The `enrollments` structure is a dynamically allocated array of ints, whose values are the lengths of the elements of the `courses` structure. In other words, for any `i`, `courses[i]` is an array of length `enrollments[i]`.

Both `courses` and `enrollments` arrays have `numCourses` elements. You can assume that `numCourses` is greater than zero, and that every element of `enrollments` is greater than 0.

In this question you will help us implement the overloaded assignment operator for the `picRosters` class.

You may assume that all pointers are valid. That is, they are either `NULL` or they point to an object of the specified type. Furthermore, you may assume that the `BMP` class has an appropriately defined "Big Three" (destructor, copy constructor, and assignment operator).

You will write your answers on the following pages. To grade the coding portions of 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. Comments will be worth up to 1/3 of the total points for any part of the problem. Adding comments to our code skeletons can get you partial credit, but it's not required.

problem 2 continued...

(a) (10 points) We have written a partial implementation for a helper member function called `copy` that makes the current object have the same *value* as the parameter `orig`.

`copy` assumes that the current object has no dynamic memory associated with it. Note that this function would be called by the `picRosters` copy constructor. Your task is to fill in the blanks so as to complete the `copy` function.

Solution:

```
void picrosters::copy(picrosters const & orig):numCourses(orig.numCourses)
{

   courses = new BMP * [numCourses];

   enrollments = new int[numCourses];

   for(int i = 0; i < numCourses; ++i)
   {

      enrollments[i] = orig.enrollments[i];

      courses[i] = new BMP[enrollments[i]];

      for(int j = 0; j < enrollments[i]; ++j)

            courses[i][j] = orig.courses[i][j];
   }
}
```

Rubric: One point for each correct blank (or equivalent code).

(b) (5 points) Write a helper member function `void clear()` that frees all memory used by an instance of `picRosters`. Note that this function would be called by the `picRosters` destructor.

Solution:

```
void picRosters::clear()
{
   for(int i=0;i<numcourses;i++)
   {
      delete[] courses[i];
   }
   delete[] courses;
   delete[] enrollments;
}
```

Rubric: 1 pt for correct header
1 pt for correct for loop structure
1 pt for each delete present and correct

-1 pt for incorrect delete or other code

(c) (5 points) We have tried to write the overloaded assignment operator for the `picRosters` class using the helper functions you wrote in parts (a) and (b). Circle the errors in our function, and write the correct code on the right, very clearly.

```
const picRoster & picRosters::operator=(const picRosters & rhs)
{

   if( this !=  &rhs)
   {
      clear();
      copy(rhs);

   }

   return *this;
}
```

Rubric: 1 pt recognizing missing return parameter picRosters const &
0.5 pt minor error in above
1 pt recognizing missing const in input param
1 pt correctly changing *this!=rhs to this!=&rhs
0.5 minor error in above
1 pt correctly switching order of copy and clear
1 pt correctly returning *this

3. **[Miscellaneous C++ – 17 points].**

Solution:

(a) (8 points) We have seen 4 different uses for the symbol ":". For each, give a short example and explain using appropriate C++ vocabulary.

i. (::) Accessing a global variable in a class or name-space.

```
std::cout
```

Defining a function - scope resolution operator

```
<return-value> <class>::<member-function>(<args>) { /* ... */ }
```

ii. (:) inheritance

```
class A : public B {
}
```

iii. (:) determining the access level

```
class A  {
    public:
// ...
    private:
//...
}
```

iv. (:) initializing a variable by a parameter - initializer list

```
void setVar(int param) : var(param) {
}
```

Rubric: 1 point for example and 1 point for the explanation.

(b) (4 points) We have seen two different uses of the keyword `const`. For each, give a short (one line) code example and briefly explain the behavior.

Solution: One point for example, one point for explanation.

i. example 1:

```
const var_type variable
```

This would create a variable of type var_type, whose value can not be changed.

ii. example 2: Const functions. Call to const functions won't change the object.

```
<return-value> <class>::<member-function>(<args>) const
{
        // ...
}
```

(c) (5 points) There are 3 characterizing components of object oriented programming among the list of concepts below. Circle them.

Solution: 2 pts for inheritance, 1.5 pts each for encapsulation and polymorphism

abstract data types	**inheritance**
classes	memory management
code re-use	parameterized types
dynamic memory allocation	**polymorphism**
encapsulation	top-down design

4. **[Inheritance – 10 points].**

In each part of this problem we will be presenting you with a pair of classes and some code snippets and asking you about the result of compiling and running the code. Your task will be to tell us explicitly what happens, and why. If there is an error, you should reason about whether the error occurs at compile time, or run time. If there is output, you should tell us what the output is. If the result depends on some other factors, explain them. Assume that the standard iostream is available (`cout` and `cin` work).

(a) (3 points) Class definitions:

```
class Rect{
public:
 Rect(int w,int h):wid(w),ht(h){}

private:
 int wid, ht;
};
```

```
class Sq:  public Rect {
public:
 Sq(int side):Rect(side,side){}

 int area(){return wid * wid;}

};
```

Code snippet:

```
Sq d(5);
cout << d.area << endl;
```

Result: **Solution:** Compiler error: class Sq cannot access Rect's private members. (1 point for an answer of 25)

(b) (3 points) Class definitions:

```
class vehicle{
public:
 void showCol(){cout << "gold";}

};
```

```
class bus:  public vehicle {
public:
 void showCol(){cout << "silver";}

};
```

Code snippet:

```
vehicle * v;
int x;

// initialize x

if (x%2 == 0)              // x is even
  v = new vehicle;
else
```

```
  v = new bus;

v->showCol();
delete v;
```

Result:
Solution: Always prints gold. (1 point for conditional response of gold/silver)

(c) (2 points)

Class definitions:

`class artifact{`
 	
`public:`
`  virtual void showMed()`
`  {cout << "gold";}`
 	
`};`
	`class basket:  public artifact {`
	
	`public:`
	`  virtual void showMed()`
	`  {cout << "clay";}`
	
	`};`

Code snippet:

```
artifact * a;
int x;

// initialize x

if (x%2 == 0)              // x is even
  a = new artifact;
else
  a = new basket;

a->showMed();
delete a;
```

Result: **Solution:** Prints gold if x is even, silver otherwise. (1 point for assuming x is even)

(d) (2 points) Class definitions:

```
class athlete{

public:
  virtual void showOff() = 0;

};
```

```
class hoopStar:  public athlete {

public:
  virtual void showOff()
    {cout << "w00t!";}
};
```

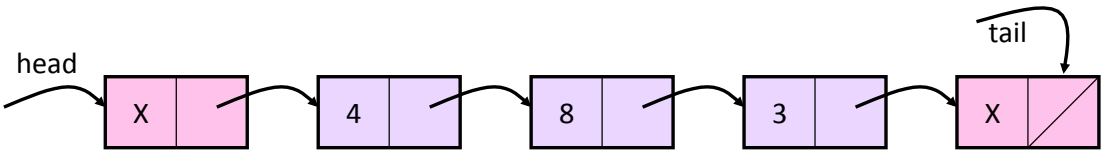
Code snippet:

```
athlete * a;
int x;

// initialize x

if (x%2 == 0)              // x is even
  a = new athlete;
else
  a = new hoopStar;

a->showOff();
delete a;
```

Result: **Solution:** Compiler error because athlete is an abstract base class (no object of that type can be constructed). (one point for printing w00t, two points for conditional printing of w00t, or runtime error.)

5. **[Lists – 33 points].**

Suppose you have implemented a List using a singly linked list with sentinels at the head and tail, and a tail pointer as in the figure below. In this problem you will implement some of the member functions for this list class.



Here is a partial `List` class definition:

```
template <class LIT>
class List{
public:
   List(const LIT & e);
   List(const List & orig);
   ~List();
   void insert(int loc, const LIT & e);
   void removeLast();
   // lots more member functions

private:
   listNode * head;
   listNode * tail;
   int size;        // the number of data elements in the list

   listNode * Find(int k, listNode * curr);
   void removeAll(listNode * curr);

   struct listNode {
     LIT data;
     listNode * next;
     listNode(LIT e): data(e), next(NULL) {} ;
     listNode(): next(NULL) {} ;
   };
};
```

The `Find` private helper function returns a pointer to the listNode that is `k` nodes beyond the input listNode *. For example, `Find( 2, head)` returns a pointer to the node containing the value 8 in the example above. You may use this helper function anywhere you'd like.

(a) (5 points) Write the default (no argument) constructor for the `List` class. To make this easy for us to grade, place the function prototype on the first line, and then write the rest of your code between the brackets. Our solution required 3 lines of code. (Note, the `next` pointer for the tail sentinel should be `NULL`.)

```
template <class LIT>
List<LIT>::List(): size(0)
{
   head = new listNode;
   tail = new listNode;
   head->next = tail.
}
```

Rubric: 1 pt - correct templating
1 pt - correct class declaration
1 pt - setting size to 0
1 pt - allocating head sentinel with the correct pointer
1 pt - allocating tail sentinel with the correct pointer

(b) (5 points) We've written part of the code for the member function `insert` below. Your task is to complete the function. `insert` puts a new `listNode` containing element `e` into the list at position `loc`. For example, in the example on the previous page, `insert(3, 2)` would place a listnode containing the value 2 into the list between the nodes containing 8 and 3.

```
template <class LIT>
void List<LIT>::insert(int loc, LIT e)
{

   listNode * t = new listNode(e);

   listNode * p = Find(loc-1, head);

   t->next = p->next;
   p->next = t
   size ++;
}
```

Rubric: 1 pt - declaring t a listNode
1 pt - using loc-1 and head (or loc and head if used properly)
1 pt - incrementing the size variable
1 pt - properly setting pointers
2 pts - properly inserting the node at the correct position
-1 for each major mistake i.e. deleting nodes (calling delete on pointers)

(c) (5 points) What is the asymptotic, worst case, running time for the best possible implementation of the member function `removeLast` that removes the last data element from

the list? Explain your answer.

To remove the last data element from the list in constant time we need a pointer to the node before that last element, or a pointer to the node itself. Since our list is singly linked, both of these strategies require that we traverse the list from the head to find the appropriate pointer. Since the list is size O(n), the remove must take O(n).

Rubric: This question was "all or nothing." I awarded 0 points for a O(1) response, and full points for O(n). Note that the tail pointer is a red herring in this problem.

(d) (5 points) Explain why the copy constructor (declared in the `List` class definition) has a constant reference parameter. Your answer should explain both the `const` and the `&`.

const: we want to assure that the VALUE of the argument to the function is not changed through its reference, and we do so by making the parameter type a reference to a constant List. (2 points)

&: (passing the List by reference) in fact, the parameter could NOT passed by value, because if it were, the copy constructor would be invoked, and that is the function we're defining! (3 points for correct soln, 1 point for explaining general reasons we might pass by reference)

(e) (3 points) Write a single line of code that invokes the default constructor you wrote in part (a).

```
List<int> a;  // OR
List<int> * a = new List<int>;
```

Rubric: 0 points for trying to CALL the constructor. The only case in which this is acceptable is in the initializer line of a derived class constructor, where the base class constructor can (and should) be invoked directly.
1 point for having something correct
2 points for using the second solution, but forgetting the `*`
2 points for List a; (forgetting the template type... this actually works in some cases.)

(f) In this part of the problem we are going to focus on writing the destructor for the `List` class. In particular, we'll write a recursive helper function that frees the memory and call that function from within the destructor.

i. (5 points) Write a recursive private helper function called `removeAll` that frees all memory allocated for the list, including the sentinel nodes. We've given you the skeleton of the function, you just have to fill it in. (We realize that we are guiding you to a particular, elegant solution. If you'd like to give a complete alternative solution, you may do so on the back of the previous page for 3 points.)

```
template <class LIT>
void List<LIT>::removeAll(listNode * curr)
{
```

```
   if (curr != NULL)
   {

      removeAll(curr->next);


      delete curr;
   }

}
```

Rubric: 2 points for "curr != NULL", 1 point for recursive call, 2 points for "delete curr".

ii. (5 points) Now write the destructor for the `List` class as it would appear in `List.cpp`. Your solution should call `removeAll` appropriately.

```
template <class T>
List<T>::~List() {
   removeAll(head)
}
```

Rubric: 1 point for template
1 point for `List <T>`
1 point for List()
2 points for removeAll(head)

scratch paper 1

scratch paper 2