

First Examination

CS 225 Data Structures and Software Principles

Fall 2010

7p-9p, Tuesday, September 28

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 20 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	20		
4	20		
5	20		
Total	100		

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

MC1 (2.5pts)

Consider the following statements, and assume the standard `iostream` library has been included:

```
int v;  
int * w;  
v = 10;  
w = v;  
*w = 8;  
cout << v << endl;
```

What is the result of executing these statements?

- (a) 8 is sent to standard out.
- (b) 10 is sent to standard out.
- (c) This code does not compile.
- (d) This code results in a runtime error.
- (e) None of these options is correct.

MC2 (2.5pts)

Consider the following statements, and assume the standard `iostream` library has been included:

```
int b;  
int * a = new int(8);  
  
b = *a;  
*a = 10;  
  
cout << b << endl;  
delete b;
```

What is the result of executing these statements?

- (a) 8 is sent to standard out.
- (b) 10 is sent to standard out.
- (c) This code does not compile.
- (d) This code results in a runtime error.
- (e) None of these options is correct.

MC3 (2.5pts)

What is the output of the following sequence of C++ statements? (The sphere class interface is included at end of the exam.)

```
sphere * a, * b;  
  
a = new sphere(1.0);  
b = a;  
b->setRadius(2.0);  
delete b;  
  
a->setRadius(4.0);  
sphere * c = new sphere(5.0);  
b = new sphere(3.0);  
  
cout << a->getRadius() << endl;
```

- (a) 4.0
- (b) 3.0
- (c) A segmentation fault.
- (d) Compiler error.
- (e) The behavior cannot be predicted.

MC4 (2.5pts)

Consider the following statements, and assume the standard `iostream` library has been included:

```
void doub(int x) { x = x * 2;}
void trip(int * x) { *x = *x * 3; }
void quin(int & x) { x = x * 5; }

int main() {

    int x = 1;

    doub(x);
    trip(&x);
    quin(x);

    cout << x << endl;

    return 0;
}
```

What is the result of executing these statements?

- (a) 1 is sent to standard out.
- (b) 5 is sent to standard out.
- (c) 6 is sent to standard out.
- (d) 10 is sent to standard out.
- (e) 15 is sent to standard out.

MC5 (2.5pts)

Which of the following correctly declares a dynamic array of Lists of pointers to strings?

- (a) `List<string> ** name;`
- (b) `List<string *> * name;`
- (c) `List * name = new string *[size]`
- (d) More than one of (a), (b), (c), are correct.
- (e) None of (a), (b), (c), are correct.

MC 6 (2.5pts)

Suppose class `stringGetter` contains exactly one pure virtual function: the overloaded parentheses operator, `string operator()(int x)`. Also suppose that class `getPageString` is a public `stringGetter` that implements `operator()`.

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

- (a) `stringGetter * a = new stringGetter;`
- (b) `stringGetter * a = new getPageString;`
- (c) `stringGetter * a;`
`getPageString * b = new getPageString;`
`a=b`
- (d) Exactly two of these will result in a compiler error.
- (e) It is possible that none of these will result in a compiler error.

MC 7 (2.5pts)

Which of the following concepts is mentioned in the “Rule of the Big Three?”

- (a) copy constructor
- (b) constructor
- (c) encapsulation
- (d) header file
- (e) None of these concepts is mentioned in the rule.

MC 8 (2.5pts)

Consider the following class definitions:

```
class Sport{
public:
    int winner() const;
private:
    int score;
};

class VolleyBall: public Sport {
public:
    int loser();
};
```

Where could the assignment `score=20;` appear for the private variable `score`?

- (a) Both `winner()` and `loser()` can make the assignment.
- (b) `loser()` can make the assignment, but `winner()` cannot.
- (c) `winner()` can make the assignment, but `loser()` cannot.
- (d) Neither `loser()` nor `winner()` can make the assignment.
- (e) The answer to this question cannot be determined from the given code.

2. [MP2ish – 20 points]. Consider the following partial class definition:

```
class RoadTrip
{
    private:
        string ** destinations;
        int duration;

        // some helper functions

    public:
        // constructors and destructor
        RoadTrip(int num); // constructor for a RoadTrip of num days

        // operator=  declaration
        // operator+  declaration

        // lots of other public member functions not relevant to this problem
};
```

The `destinations` structure is a dynamically allocated array of `string` pointers. The array `destinations` has `duration` elements. A place name (of type `string`) is added to the destination structure in the cell whose number corresponds to the first day of arrival at that place. We assume that cells with no destination are simply layovers in the most recent place, and that day 0 contains our starting location. For example, if we start in Urbana, arrive in Milwaukee on the first day, and end in Lincoln on the fifth day, then `destinations[0] == "Urbana"`, `destinations[1] == "Milwaukee"`, `destinations[5] == "Lincoln"`. In this example, the `duration` variable should be 6. You can assume that `duration`, when it is specified, is greater than zero. You can also assume the `string` class has been included and scoped.

You may assume that all pointers are valid. That is, they are either `NULL` or they point to an object of the specified type. In particular, the `RoadTrip(int num)` constructor builds a dynamic array of length `num`, whose elements are all `NULL`, and sets `duration` to `num`.

In this question you will help us implement some of the member functions for the `RoadTrip` class.

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) (8 points) In this part of the problem, you will write the code for an overloaded addition operator, so that two `RoadTrips` can be added together. The addition of two road trips is a simple concatenation of the days of the trips into one larger adventure. Specifically, If road trips `m` and `n` are declared like: `RoadTrip m(10), n(20);`, then `m + n` is a road trip of length 30 consisting of all the days of `m` followed by all the days of `n`. Recall that some days might be `NULL`. These should be preserved in the `RoadTrip` sum.

```
----- RoadTrip::-----(----- rhs)
{

    RoadTrip m(-----);

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

        if (----- != NULL) {

            ----- = -----;

            -----;
            // this line ^^^ depends on the previous line, and may be blank
        }
    }

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

        if (----- != NULL) {

            ----- = -----;

            -----;
            // this line ^^^ depends on the previous line, and may be blank
        }
    }
    return -----;
}
```


(b) (6 points) Write the destructor for the `RoadTrip` class.

(c) (6 points) The following function is intended to take one destination in a `RoadTrip` and move it to a different location within the same trip. Specifically, the contents of `destinations[target]` are to be replaced by the contents of `destinations[source]`, which should then be left empty, or `NULL`. Unfortunately, the code below doesn't behave the way we intend. Please complete and correct the function. Your code will be partially graded on efficiency (don't copy data if you don't need to). You may assume that `source` and `target` are valid indices, though you cannot assume that their contents are non-`NULL`.

```
void RoadTrip::changeDestination(int source, int target)
{

    *(destinations[target]) = *(destinations[source]);

}
```

3. [MP3ish – 20 points].

The following code is a partial definition of a doubly linked list implementation of the `List` class that you used for MP3. Note in particular that it does not contain sentinels, but it *does* have head and tail pointers.

```
template <typename Etype>
class List {
public:

    // splitList
    //   - parameters : rank - an integer; the rank-th node of the list
    //                   is the first node of the split-off list
    //   - returns the portion of the current list from
    //         the rank-th node onward; the current list is reduced
    //         to the portion occurring before the rank-th node
    List<Etype> splitList(int rank);

    // a bunch of other List class functions

private:

    class ListNode
    {
    public:
        // ListNode constructor
        //   - initializes element to default Etype, and pointers to NULL
        ListNode();

        // ListNode constructor
        //   - parameters : value - the value to store in the element field
        //   - initializes node to hold value and NULL pointers
        ListNode(Etype const & value);

        // Maybe some other functions here.

        ListNode* next; // pointer to next node in list
        ListNode* prev; // pointer to prior node in list
        Etype element;  // holds element of node
    };

    ListNode* head; // points to first node of list
    ListNode* tail; // points to last node of list
    int size;
};
```

- (a) (10 points) Please implement the `splitList` function (as you did for MP3) This function finds the rank-th element of the list, and detaches the portion of the list starting there from the preceding part of the list. The detached portion is then placed in a new list. If rank is greater than size, then return an empty list (leaving the original intact). The possible values for rank are integers greater than or equal to 1. That is, You can assume that the integer argument rank is positive. Note that the node pointed to by the List class' head pointer is considered to be the "first" node, and not the "zeroeth". For this function, you may declare a single variable of type `List<Etype>`, for the sole purpose of holding the list that is returned to the calling function. However, you are not allowed to create any new nodes in this function – you can only manipulate the existing nodes.

Be sure to make your code very neat and easy to read. Draw pictures where it will help us, and comment profusely.

(b) (5 points) Give a tight worst case upper bound for the running time of your implementation of `splitList` if the list contains n items.

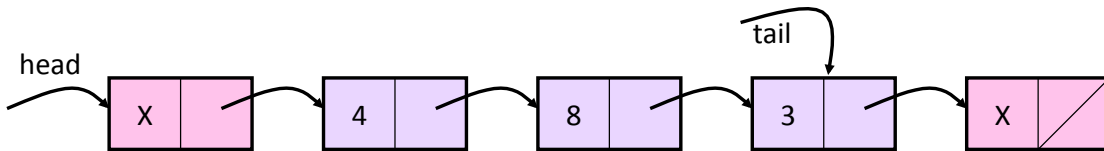
(c) (3 points) Does the `ListNode` class require a destructor? Briefly justify your answer.

(d) (2 points) Explain why the `ListNode` class is private within the `List` class.

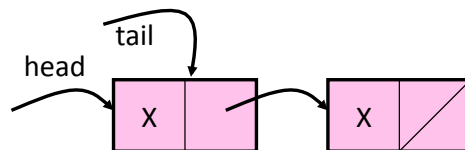
4. [Links – 20 points].

Suppose you have implemented a List using a singly linked list with sentinels at the head and tail, and a tail pointer pointing to the node *before* the tail sentinel node. In the general case, this corresponds to the last data item in the list as in the figure below, and in an empty list, it corresponds to the head sentinel. In this problem you will implement and analyze some of the member functions for this list class.

A list with 3 elements:



A list with 0 elements:



Here is a partial List class definition (continued on next page):

```

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;
    listNode * curr;
    int size;          // the number of data elements in the list

    listNode * Find(int k, 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 above the first line, and then write the rest of your code between the brackets. (Note: the `next` pointer for the tail sentinel should be `NULL`.)

```
-----  
{
```

```
}
```

- (b) (5 points) We've written part of the code for the member function `removeLast` below. Your task is to complete the function. `removeLast()` removes the last data item in the list if the list is not empty, and does nothing, otherwise. In the non-empty example on the previous page, `removeLast()` would remove the node containing value 3.

```
template <class LIT>  
void List<LIT>::removeLast()  
{
```

```
}
```

- (c) (8 points) Complete the table below with tight asymptotic running times (using big-O notation) for the following `List` class functions on data of size n . In the table we compare the linked memory implementation of this problem with the best array-based implementation we can imagine. You may assume that the space in the array is sufficient for all list operations.

	Singly linked list	Array
<code>insertAtFront(const LIT & e)</code>		
<code>insertAtRear(const LIT & e)</code>		
<code>removeFront()</code>		
<code>removeRear()</code>		

- (d) (2 points) Explain the role of the tail sentinel node in the functions listed in the table above. Would the running times of these functions change if there were no sentinel?

5. [Miscellaneous – 20 points].

- (a) (5 points) We have tried to write the overloaded assignment operator for the `Food` class, but our code isn't working! Assume we have tested the `copy` and `clear` functions, and they do what they're supposed to do. Circle the errors in our function, and then write the correct code on the right, very clearly.

```
Food::operator=(Food & rhs){  
  
    if( *this != rhs) {  
        copy(rhs);  
        clear();  
    }  
  
    return this;  
}
```

Now, answer the 3 questions below based on your corrected code.

- i. (2 points) Explain the type specification in the parameter list. In particular, why is the parameter passed by reference?

 - ii. (2 points) Justify the conditional (`if`) by explaining its purpose.

 - iii. (2 points) Explain the return value. What is its purpose?
- (b) (2 points) Briefly describe a situation in which we would choose to pass an object by reference, rather than by value. Please give an example different from the one in the previous problem.
- (c) (2 points) Briefly describe one scenario in which we would choose to pass an object by reference, over passing a pointer to the object by value. Please give a reason that is more substantive than mere syntactic simplicity.

(d) (2 points) Briefly describe two instances when the copy constructor is invoked.

-

-

- (e) (1 point) On a scale of 1 to 5, how much are you learning in the class? (1 is not much, 5 is a ton)
- (f) (1 point) On a scale of 1 to 5, how is the pace of the course so far? (1 is too fast, 5 is too slow)
- (g) (1 point) 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.

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

    double getDiameter() const;
    double getRadius() const;
    void setRadius(double r);
    void setPictureGetRadius(BMP & newPicture);

private:
    double theRadius;
    BMP thePicture;
};
```

scratch paper 1

scratch paper 2