Com S 228 Introduction to Data Structures

Exam 1

6:30-7:30pm
Thursday, Feb 17, 2005

Name: ________________________________

ID (4-digit): ___ ___ ___ ___

Section: _____

No calculator is allowed.
To ensure fairness, everyone will be given the same amount of time, so you must stop writing when time is called. If you do not stop writing when instructed, you will receive a 25% deduction in your score.
<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24</td>
<td>14</td>
<td>16</td>
<td>21</td>
<td>25</td>
</tr>
</tbody>
</table>
1. [24 pts] Short Questions

(a) [8 pts] Determine if the following statements are true or false. For each statement, mark only the answer you think is correct.

(i) A friend function of a class has access to its private data members.

true ______ false ______

(ii) When the operator “+” is overloaded as a member function, the implicit object is the rightmost operand.

true ______ false ______

(iii) C++ always provides a default constructor to a class if it does not have one.

true ______ false ______

(iv) If you don’t write a copy constructor for a class, the compiler provides one that does shallow copying.

true ______ false ______

(b) [4 pts] What is the difference between private data members and public data members of a class?
(c) [4 pts] Give a situation in which copying will be involved but the copy constructor will not.

(d) [4 pts] What is a memory leak?

(e) [4 pts] What is the value of *this inside the implementation of a class member function?
2. [14 pts] Arrays, Pointers, and References

Consider the five variables declared below:

```c
int a[] = { 5, -2, 7, 8};
int &b = a[0];
int *c = &(a[3]);
int **d = c;
int **e = &d;
```

(a) [3 pts] Which variables are pointers?

(b) [3 pts] Which variables are references?

(c) [8 pts] Fill the elements in the array `a` (drawn below) after the sequential execution of these six statements:

```c
b--;
*d += b + a[1];
c = &(a[1]);
**e -= a[2];
c[0] = *d + **e + b;
d[1] = 2 * (**e);
```

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3
3. **[16 pts] Class and Dynamic Memory**

You are provided a class Dynamic to store a pair of integers.

```cpp
class Dynamic
{
public:
    Dynamic(int m = 1, int n = 1);
    Dynamic(const Dynamic& d);
    ~Dynamic();

private:
    int *pair;
};

Dynamic::Dynamic(int m, int n)
{
    pair = new int [2];
    pair[0] = m;
    pair[1] = n;
}

Dynamic::Dynamic(const Dynamic& d)
{
    pair = new int [2];
    pair[0] = d.pair[0];
    pair[1] = d.pair[1];
}

Dynamic::~Dynamic()
{
    delete [ ] pair;
}
```

Inside the two functions main and createAnObject below, comment on all invocations of the constructors and the destructor of the class Dynamic. You should add your comments to the right of every statement (except a function call) whose execution either involves or will immediately trigger such an invocation. Follow the rules below when writing your comments.

- If an object, say, foo, is created with its private data member having values pair[0] = 1 and pair[1] = -1, simply write “object foo(1, -1) constructed”.

4
• If an object, say, foo, is destructed, write “object foo destructed”.
• If a heap (or free store) object is created or destructed, refer to it by any of its pointers, say, ptr. And write something like “a heap object constructed and pointed to by ptr” or “the heap object which ptr points to is destructed”.
• If no object is created after execution of the statement, leave the comment space blank.

```cpp
void createAnObject(int m, int n)
{
    Dynamic a(m, n); //
    //
}

void main()
{
    Dynamic b(2); //
    //
    Dynamic *p = new Dynamic(3, 3); //
    //
    Dynamic *q = p; //
    //
    Dynamic e = *p; //
    //
    createAnObject(2, 3);
    p = &e; //
    //
    delete q; //
    //
}
```
4. 21 pts Linked Lists

(a) 8 pts Consider the linked list class in Homework 3.

```cpp
class LinkedList
{
    public:
        ...

    bool ListEmpty() const;
    // returns true if the list is empty and false otherwise

    void Reset(int pos = 0);
    // reset the current position to pos

    void InsertFront(const int& item);
    // inserts a node with data value item at the front
    // of the list

    void InsertRear(const int& item);
    // inserts a node with data value item at the rear
    // of the list

    int DeleteFront();
    // deletes the front node from the list and returns
    // the value of its data item

    ...
};
```

You are given the following declarations

```cpp
LinkedList L1;
LinkedList L2;
```

Assume that the linked list L2 contains, from its front to rear, integers 2, -11, 8, 93, -71, 66, 13. Also assume that the linked list L1 is empty. The following statements are executed.
L2.Reset(1);
while (!L2.ListEmpty())
{
    L1.InsertFront(L2.DeleteFront());
    if (!L2.ListEmpty())
        L1.InsertRear(L2.DeleteFront());
}

Describe the contents of each list as a sequence after the above execution.

L1:

L2:

(b) [13 pts] This problem assumes that linked lists are created and maintained using the Node class.

    class Node
    {
        public:
            int data;

            // constructor
            Node (const int& item, Node* ptrnext = NULL);

            // insert a node after the current node
            void InsertAfter(Node *p)
            { p->next = next; next = p; }

            // obtain the address of the next node
            Node *NextNode() const { return next; }

        private:
            Node *next;
    };

The last node in a linked list always has its private data member next == NULL.
Write the function

    void MergeLists(Node* L1, Node* L2, Node* &L3);

to merge two existing lists with the header pointers L1 and L2, respectively, into a third list with the head pointer L3. The elements in each of the two input lists are distinct and already in the increasing order. The elements in the output list L3, which is empty before the call, will assume the increasing order. Be sure to eliminate duplicates in L3.
5. [25 pts] Operator Overloading

On the last two pages of the exam (which you can tear off) is the header file for a class that handles answers to a survey question. Survey questions can collect either numerical data (like age or GPA) or categorical data (like eye color or preferred soda brand), so this class has to contain a dynamic array for numbers and another for strings. However, since a given question will produce only one kind of data, the class has a flag which is 'N' for numbers or 'C' for categories. The array pointer for the other kind of data will be set to NULL.

Before answering the two questions below, read the class declaration carefully.

(a) [11 pts] In the space below, write the implementation of the assignment operator for class SurveyAnswers, as it would appear in the .cpp file.
(b) [14 pts] In the space below, write the implementation of the output operator for class SurveyAnswers, as it would appear in the .cpp file.
class SurveyAnswers
{
public:
    SurveyAnswers(char dataType);
    //Pre:  dataType is 'N' or 'C'
    //Post: The dataType is stored in the flag, the number of answers is
    //       zero, and both arrays are empty

    ~SurveyAnswers();
    //Post: memory is deallocated

    SurveyAnswers(const SurveyAnswers& other);
    //Post: A deep copy constructor

    SurveyAnswers& operator=(const SurveyAnswers& other);
    //Post: A deep copying assignment operator, with the usual value
    //       semantics.

    void readData(string filename);
    //Post: The number of answers is read from the file and the appropriate
    //       type of data answers are read into either the numericalAns or
    //       catagoricalAns array.

    void sort();
    //Pre: The readData function has been called.
    //Post: Either the numericalAns or catagoricalAns array is sorted into
    //       increasing order

    friend ostream& operator<<(ostream& oStr, SurveyAnswers s);
    //Pre:  The readData function has been called.
    //Post: If the flag is 'N', then the number of answers, the minimum, the
    //       maximum, and the average answer is printed to oStr. If the flag
    //       is 'C', then the number of answers, and the catagory answers and
    //       the number of times each answer appeared in the results will be
    //       printed to oStr. (The answers to the catagorical question do not
    //       have to be printed in any particular order.)
    //       The return value is oStr.
    //Example:
    //   If the SurveyAnswers object gpa contains 34 answers about
    //   people’s GPAs, then cout << gpa might print something like:
    //   Number of Answers: 34 Minimum: 0.86 Maximum: 3.95 Average: 2.4
    //   If the SurveyAnswers object colaBrands contains 7 answers,
    //   then cout << colaBrands might print something like:
    //   Number of Answers: 7 Coke: 3 Pepsi: 2 Sierra Mist: 2
private:
  char flag; // Either 'C' or 'N'
  double* numericalAns; // If flag is 'N', then points to a dynamic array,
                         // otherwise points to NULL.
  string* catagoricalAns; // If flag is 'C', then points to a dynamic array,
                          // otherwise points to NULL.
  int numAnswers;
};