Computer Science III (22C:30, 22C:115)
Exam 1 September 27th 2002

The exam is worth 150 points (15% of your total grade). Feel free to refer to your textbook, class-notes, or printouts of any of the sample programs posted on the course page. You have 50 minutes to complete the exam.

Problem 1 [55 points]
Here is the header for the node class. This was discussed in class in the context of linked lists.

```
class node{
      public:
             node(int x, node* next);
             node* getlink();
             void setlink(node* next);
             void setdata(int x);
      private:
             int myData;
             node* link;
};
```

(a) [**10 points**] Does this class need a copy constructor? Explain in a couple of sentences.

(b) [**25 points**] Write the implementation of a copy constructor for the node class.

(c) [**20 points**] Here is a little program that uses the node class. Draw a picture of the linked list that node pointer `b` is pointing to, (a) just before the call to `foo` and (b) just after the call to `foo`.

```
#include <cstdlib>
#include "node.h"


void foo(node t){
     node* a;
     a = t.getlink();
     (*a).setdata(10);
}


int main(){
    node* b;
    b = new node(20, NULL);
    node* c;
    c = new node(30, NULL);
    (*b).setlink(c);

    foo(*b);

    return EXIT_SUCCESS;
}
```

Problem 2 [55 points]
Suppose we want to define a class called `NameList` that maintains a list of names. Further suppose that the data members of this class are defined as:

```
apvector<apstring> myNames;
int mySize;
```

The `int` variable `mySize` keeps track of the number of names in the list.

Suppose we want to overload the `+=` operator so that we can use it to "join" two lists of names. Specifically, we would like code such as the following to work so that after the `L1 += L2` statement is executed, `L1` contains all the names that it originally contained, along with all the names `L2` contained. Multiple occurrences of the same name in the list is okay and the order in which names occur in the list does not matter.

```
NameList L1;
NameList L2;
    ...
    ...
    ...
L1 += L2;
```

(a) [**25 points**] Write the implementation of the function that overloads the `+=` operator. Use the following function header:

```
void NameList::operator +=(const NameList & r)
```

(b) [**10 points**] Explain in one sentence why the above function does not support joining of three lists at a time, as follows.

```
L1 += L2 += L3;
```

(c) [**10 points**] Modify the implementation in Part(a) to allow for the multiple join shown in Part (b). You may have to change the function header also.

(d) [**10 points**] Does your function in Part (c) allow for a list to be joined to itself, as follows? Explain in one or two sentences.

```
L1 += L1;
```

Problem 3 [20 points]
Explain in two or three sentences, why we would want to write functions that return a reference. Use examples we have seen (either in class or in the textbook) to illustrate your answer.

Problem 4 [20 points]
On Page 180 (Figure 4.11) of the textbook you will see the implementation of the `insert` member function of the `bag` class. While the function is correct, it can lead to a somewhat inefficient implementation of the `bag` class. Explain in one or two sentences why this is so. Also, suggest a simple modification that fixes this problem.