Computer Science III (22C:30, 22C:115)
Exam 1 September 26th 2003

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.

Here is the header for a `matrix` class that can be used to define 2-dimensional integer arrays.

```
class matrix{
      public:
             matrix(int r, int c);
             ~matrix();
      private:
             int** myMatrix ;
             int numRows;
             int numCols;
};
```

1. The constructor `matrix(int r, int c);` constructs a 2-dimensional array with `r` rows and `c` columns. Write down the implementation of this constructor.

2. Here is an attempt at implementing the destructor for the class.

   ```
   matrix::~matrix(){
        delete [] myMatrix;
   }
   ```

   Explain in one sentence what is wrong with this implementation.

3. Reimplement the destructor for the `matrix` class correctly.

4. Here is an attempt at overloading the `[ ]` operator.

   ```
   int* matrix::operator [](int i){
           assert(i < numRows);
           return myMatrix[i];
   }
   ```

   Assume that the `[ ]` operator is overloaded as above and defined as part of the `matrix` class. Now consider the following piece of code. How many times is the above function called by the following piece of code? Will this code correctly assign the value 500 to the slot in row 5, column 7 in `X` and then send it to the output (yes or no)?

   ```
   matrix X(10, 30);
   X[5][7] = 500;
   cout << X[5][7] << endl;
   ```

5. Recall that we defined a 2-dimensional array in class by using a vector of vectors. Even in that example, we could use `X[i][j]` to access a particular slot in the array. In our current example of the `matrix` class, if we overload the `[]` operator as shown above, it is possible to use `X[i][j]` to access particular slots, but there is one major problem with this implementation that was not present when we used the vector of vectors implementation. What is this problem?

6. Implement a copy constructor for the `matrix` class. I am providing the function header below.

```
matrix::matrix(const matrix & rhs)
```

7. A student comes to my office and asks for help overloading the = operator for the matrix class. I start by using the following header:

```
void matrix::operator =(const matrix & rhs)
```

   Then I say, "The code that goes inside this function is identical to what we would write for the copy constructor, except that we have to perform two additional tasks in the implementation of the = operator, that were unnecessary in the copy constructor." Explain in one sentence each what these additional tasks are. If you feel more comfortable showing the implementation of = with the additional tasks highlighted, feel free to do that.

8. The implementation of the = operator described above, will not permit the following code to work correctly. Explain in one sentence what changes we have to make to the implementation so that the following code works fine.

```
matrix X(10, 20);
matrix Y(15, 30);
matrix Z(30, 30);
X = Y = Z;
```

9. Assume that the `matrix` class is templated so that instead of just integer type, the slots can have an arbitrary type. Write a code fragment that constructs an object `X` that is a 10×20 matrix of integer vectors, each vector of length 100.

10. Given the current definition of the `matrix` class, it does not seem possible to construct an object `X` that is a vector of length 100, each slot in the vector containing an 10×20 matrix. What member function would you add to the `matrix` class to make this possible? You don't have to implement this function, just mention its name and explain in one sentence what it does.
