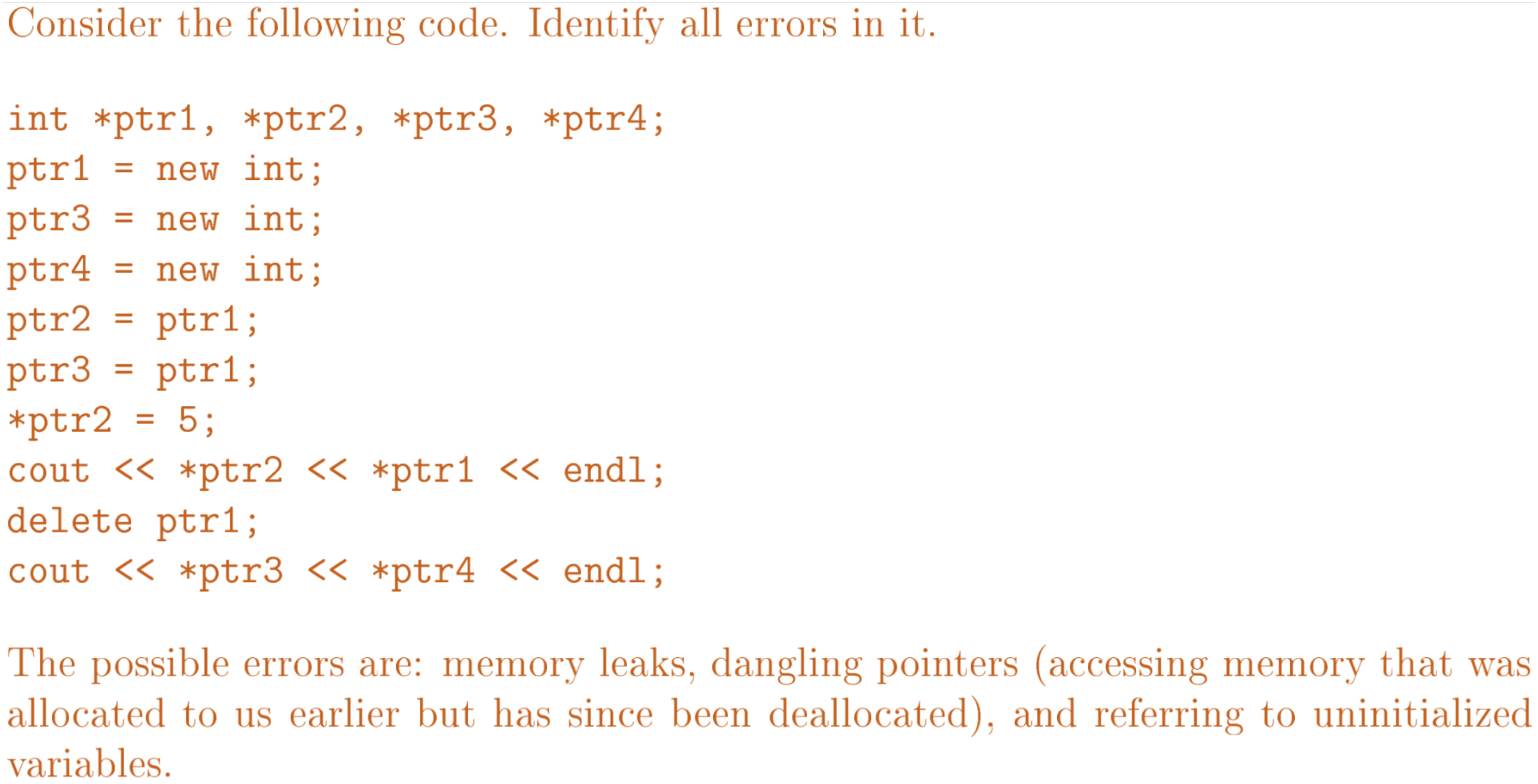
Practice Examples for Lab: Set 16

- 1



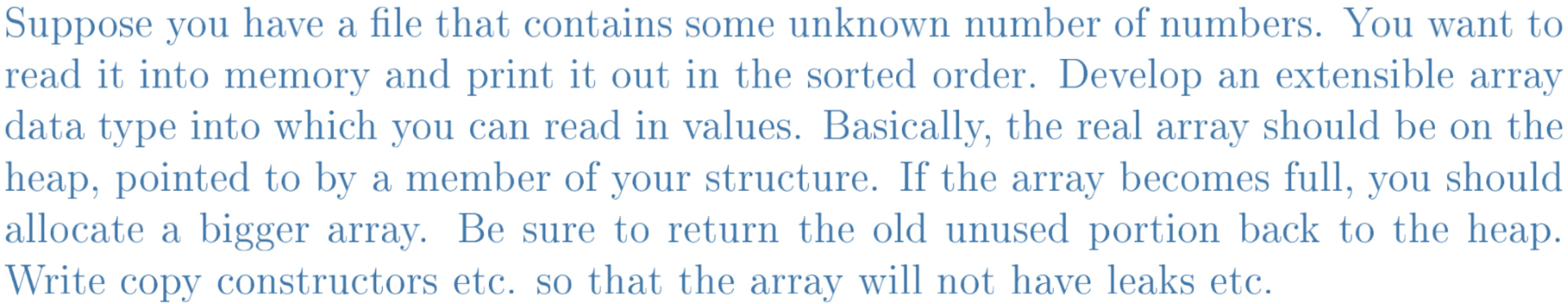
Consider the following code. Identify all errors in it.

```
int *ptr1, *ptr2, *ptr3, *ptr4;
ptr1 = new int;
ptr3 = new int;
ptr4 = new int;
ptr2 = ptr1;
ptr3 = ptr1;
*ptr2 = 5;
cout << *ptr2 << *ptr1 << endl;
delete ptr1;
cout << *ptr3 << *ptr4 << endl;
```

The possible errors are: memory leaks, dangling pointers (accessing memory that was allocated to us earlier but has since been deallocated), and referring to uninitialized variables.

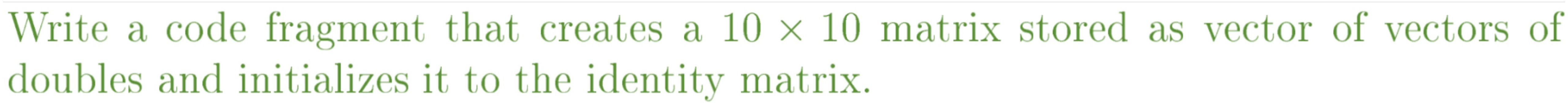
Practice Examples for Lab: Set 16

- 2



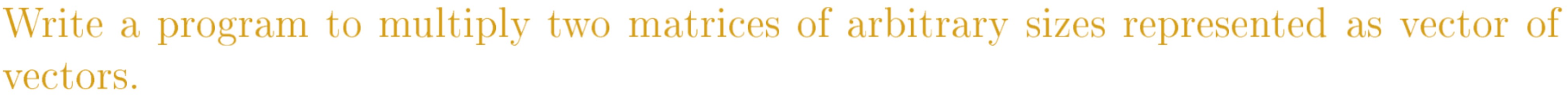
Suppose you have a file that contains some unknown number of numbers. You want to read it into memory and print it out in the sorted order. Develop an extensible array data type into which you can read in values. Basically, the real array should be on the heap, pointed to by a member of your structure. If the array becomes full, you should allocate a bigger array. Be sure to return the old unused portion back to the heap. Write copy constructors etc. so that the array will not have leaks etc.

- 3



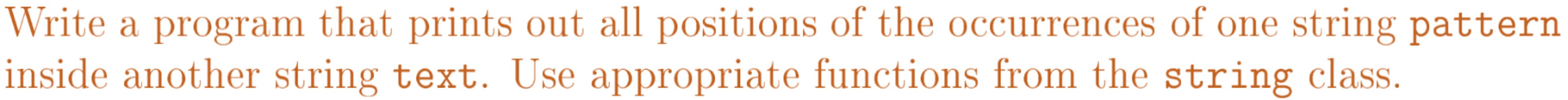
Write a code fragment that creates a 10 × 10 matrix stored as vector of vectors of doubles and initializes it to the identity matrix.

- 4



Write a program to multiply two matrices of arbitrary sizes represented as vector of vectors.

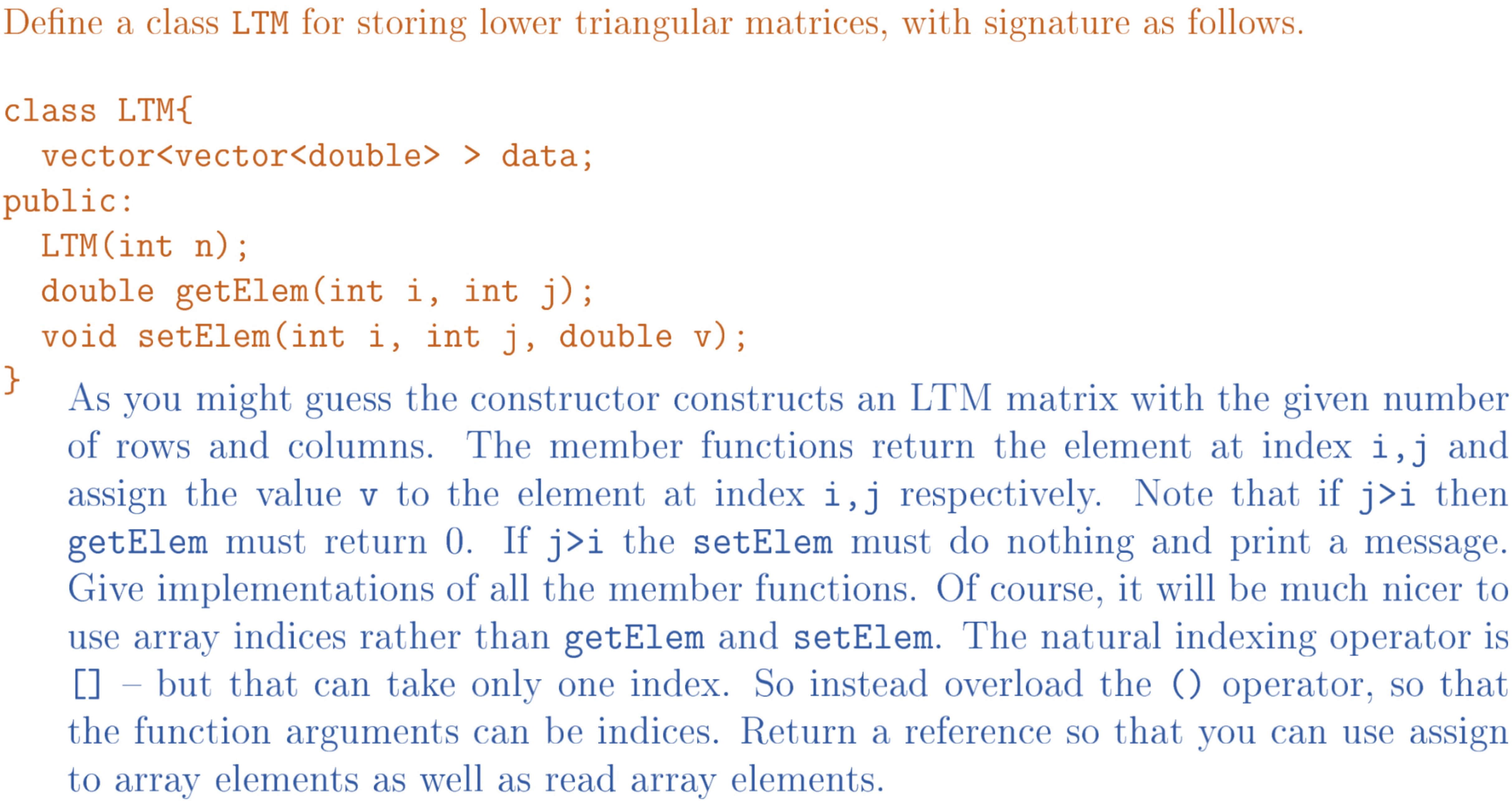
- 5



Write a program that prints out all positions of the occurrences of one string `pattern` inside another string `text`. Use appropriate functions from the `string` class.

Practice Examples for Lab: Set 16

- 6



Define a class LTM for storing lower triangular matrices, with signature as follows.

```
class LTM{
  vector<vector<double> > data;
public:
  LTM(int n);
  double getElem(int i, int j);
  void setElem(int i, int j, double v);
}
```

As you might guess the constructor constructs an LTM matrix with the given number of rows and columns. The member functions return the element at index i,j and assign the value v to the element at index i,j respectively. Note that if j>i then getElem must return 0. If j>i the setElem must do nothing and print a message. Give implementations of all the member functions. Of course, it will be much nicer to use array indices rather than getElem and setElem. The natural indexing operator is [] – but that can take only one index. So instead overload the () operator, so that the function arguments can be indices. Return a reference so that you can use assign to array elements as well as read array elements.