4. A window is represented by an *M*-by-*N* matrix filled with integers representing colors. Operations on a window include the following.

- Determine if a point lies within the window.
- Place a square of a single color in the window, ignoring those points in the square that are not within the window.

Consider the following declarations for `Window`.

```
class Window
{
  public:

    // ... constructors not shown

    bool IsInBounds(int row, int col) const;
    // postcondition: returns true if the point (row, col) is
    //                in this window;
    //                otherwise, returns false

    void ColorSquare(int ULrow, int ULcol, int N, int val);
    // postcondition: all points in this window that are also in the
    //                N-by-N square with upper left corner
    //                (ULrow, ULcol) have been set to val;
    //                points in the square that are not in this
    //                window are ignored

    int ValAt(int row, int col) const;
    // postcondition: returns color value at position row, col
    //                in this window

    // ... other public member functions not shown

  private:
    int myNumRows;
    int myNumCols;
    apmatrix<int> myMat;
};
```

(a) Write the `Window` member function `IsInBounds`, as started below. `IsInBounds` checks whether a single point is in the window.

For example, for any 5-by-4 `Window W`, the following table shows the results of several calls to `IsInBounds`.

Call	Return value
`W.IsInBounds(0, 0)`	`true`
`W.IsInBounds(2, 1)`	`true`
`W.IsInBounds(4, 3)`	`true`
`W.IsInBounds(5, 3)`	`false`
`W.IsInBounds(3, -1)`	`false`
`W.IsInBounds(8, 8)`	`false`

Copyright © 2001 by College Entrance Examination Board. All rights reserved.
Advanced Placement Program and AP are registered trademarks of the College Entrance Examination Board.

GO ON TO THE NEXT PAGE.

Complete function `IsInBounds` below.

```
bool Window::IsInBounds(int row, int col) const
// postcondition: returns true if the point (row, col) is
//                in this window;
//                otherwise, returns false
```

(b) Write the `Window` member function `ColorSquare`, as started below. `ColorSquare` sets all the integers in a specified square to a particular color value. `ULrow` and `ULcol` specify the location of the upper left corner of the square, `N` is the number of rows and columns in the square, and `val` is the color value. Points that are in the specified square but do not lie in the `Window` are ignored.

For example, consider the following 5-by-6 `Window W`.

```
10 10 10 10 10 10
10 10 10 20 20 20
20 20 30 30 30 30
30 30 40 40 40 40
40 40 50 50 50 50
```

After the call `W.ColorSquare(2, 1, 3, 66)`, `W` is changed to

```
10 10 10 10 10 10
10 10 10 20 20 20
20 66 66 66 30 30
30 66 66 66 40 40
40 66 66 66 50 50
```

After an additional call, `W.ColorSquare(2, 4, 3, 77)`, `W` is changed to

```
10 10 10 10 10 10
10 10 10 20 20 20
20 66 66 66 77 77
30 66 66 66 77 77
40 66 66 66 77 77
```

Note that the third column of the square added is not in `W`.

In writing function `ColorSquare`, you may call function `IsInBounds` specified in part (a). Assume that `IsInBounds` works as specified, regardless of what you wrote in part (a).

Complete function `ColorSquare` below.

```
void Window::ColorSquare(int ULrow, int ULcol, int N, int val)
// postcondition: all points in this window that are also in the
//                N-by-N square with upper left corner
//                (ULrow, ULcol) have been set to val;
//                points in the square that are not in this
//                window are ignored
```

Copyright © 2001 by College Entrance Examination Board. All rights reserved.
Advanced Placement Program and AP are registered trademarks of the College Entrance Examination Board.

GO ON TO THE NEXT PAGE.

(c) A rectangular area in a window can be specified using the Rectangle structure as declared below.

```
struct Rectangle
{
  int ULrow;    // row position of upper left corner of rectangle
  int ULcol;    // column position of upper left corner of rectangle
  int numRows; // number of rows in rectangle (height)
  int numCols; // number of columns in rectangle (width)
};
```

The following example shows a 5-by-4 window in which the 3-by-2 rectangle with upper left corner (2,1) is highlighted.

10	20	30	40
10	20	30	40
10	**99**	**55**	40
10	**44**	**33**	40
10	**77**	**66**	40

Write the free function Enlarge, as started below. Enlarge magnifies a Rectangle in the Window by replacing each point with a factor-by-factor square of points of the same color. The upper left corner of the magnified Rectangle is the same as the upper left corner of the original Rectangle. Each square of color is placed in the Window at the same relative position in the magnified Rectangle as the original point in the Rectangle. Conceptually, the enlarged rectangle may run off the window, but only points in the window are modified by Enlarge.

For example, consider the 10-by-11 Window W, and Rectangle R, where R.ULrow = 2, R.ULcol = 1, R.numRows = 2, and R.numCols = 4. The following table shows the original version of W with R highlighted and the result of magnifying R in W by a factor of 3.

Window W with R highlighted

00	00	00	10	10	10	10	10	10	10	10
10	10	10	20	20	20	20	20	20	20	20
20	**55**	**99**	**33**	**66**	20	20	20	30	30	30
30	**22**	**88**	**77**	**44**	30	30	30	40	40	40
40	40	40	40	30	30	30	50	50	50	50
50	50	50	40	40	40	40	30	30	30	30
60	60	50	50	50	40	40	40	30	30	20
70	70	70	50	50	50	40	40	40	30	30
80	80	70	70	60	60	50	50	40	40	30
90	80	70	60	60	50	50	40	40	30	20

Result of the call Enlarge(W, R, 3)

00	00	00	10	10	10	10	10	10	10	10
10	10	10	20	20	20	20	20	20	20	20
20	**55**	**55**	**55**	**99**	**99**	**99**	**33**	**33**	**33**	**66**
30	**55**	**55**	**55**	**99**	**99**	**99**	**33**	**33**	**33**	**66**
40	**55**	**55**	**55**	**99**	**99**	**99**	**33**	**33**	**33**	**66**
50	**22**	**22**	**22**	**88**	**88**	**88**	**77**	**77**	**77**	**44**
60	**22**	**22**	**22**	**88**	**88**	**88**	**77**	**77**	**77**	**44**
70	**22**	**22**	**22**	**88**	**88**	**88**	**77**	**77**	**77**	**44**
80	80	70	70	60	60	50	50	40	40	30
90	80	70	60	60	50	50	40	40	30	20

In writing Enlarge, you may call any public Window member functions. Assume that IsInBounds and ColorSquare work as intended, regardless of what you wrote in parts (a) and (b).

Complete function Enlarge below.

```
void Enlarge(Window & W, const Rectangle & rect, int factor)
// precondition:  factor > 0
```

END OF EXAMINATION

Copyright © 2001 by College Entrance Examination Board. All rights reserved.
Advanced Placement Program and AP are registered trademarks of the College Entrance Examination Board.