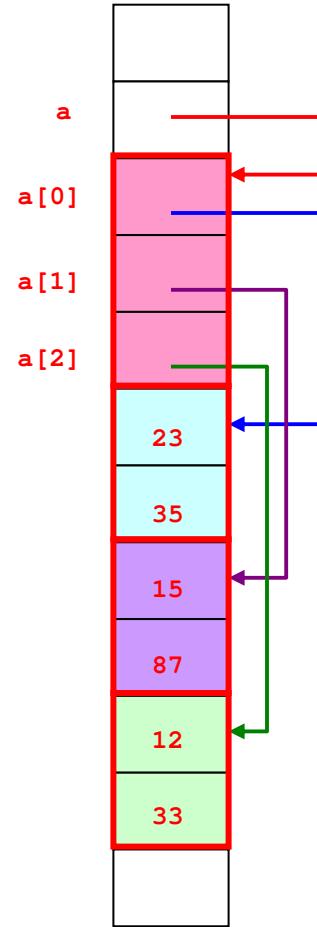


1. Draw this array as it is laid out in memory. Don't forget the row pointers **a[0]**, **a[1]**, and **a[2]**, and the pointer to the array of row pointers, **a**. Also, write the value of each array element itself (**a[i][j]**) where it is stored in memory.

```
int a[3][2] = { {23, 35},
                {15, 87},
                {12, 33} };
```



2. Here is a function prototype:

```
int rowSum( int row, int A[ ][ MAXCOLS ], int nc );
```

This function returns the sum of the elements in one row of a 2D array. The row to be summed is given as the first parameter. The array itself is given as the second parameter. The array has at most MAXCOLS number of columns. The number of columns that are actually used is given as the last parameter. Write the definition of this function.

```
int rowSum( int row, int A[ ][ MAXCOLS ], int nc )
{
    int sum = 0;
    for( int c = 0; c < nc; c++ )
        sum = sum + A[row][c];
    return sum;
}
```

3. Here is a function prototype:

```
int colSum( int col, int A[ ][ MAXCOLS ], int nr );
```

This function returns the sum of the elements in one column of a 2D array. The column to be summed is given as the first parameter. The array itself is given as the second parameter. The array has at most MAXCOLS number of columns. The number of rows that are actually used is given as the last parameter. Write the definition of this function.

```
int colSum( int col, int A[ ][ MAXCOLS ], int nr )
{
    int sum = 0;
    for( int r = 0; r < nr; r++ )
        sum = sum + A[r][col];
    return sum;
}
```

4. Here is a function prototype:

```
bool winner( int A[3][3] );
```

It is known in advance that A is a 3x3 array, and all elements of the array are being used (i.e., the entire array is filled in). This function returns **true** if any row or column sum is equal to 3 or equal to 6 ... otherwise it returns **false**. Write the definition of this function.

```
bool winner( int A[3][3] )
{
    for( int i = 0; i < 3; i++ )
    {
        int rsum = rowSum( i, A, 3 );
        int csum = colSum( i, A, 3 );
        if( rsum == 3 || csum == 3 || rsum == 6 || csum == 6 ) return true;
    }

    return false;
}
```

5. Complete the code below to (1) dynamically allocate a 20×12 array of integers named "board", (2) de-allocate it.

```
const int NROW = 20;
const int NCOL = 12;

int main()
{
    int **board;
    // (1) Dynamically allocate 'board', a 20 x 12 array:

    board = new *int[ NROW ];
    for( int i = 0; i < NROW; i++ )
        board[ i ] = new int[ NCOL ];

    // (2) De-allocate the 20 x 12 array named 'board':

    for( int i = 0; i < NROW; i++ )
        delete [] board[ i ];
    delete [] board;

    return 0;
}
```

6. Write the code that would initialize the 'board' array such that every element is zero (0), except for the first and last columns and the last row, which are initialized to one (1).

```
for( int r = 0; r < NROW; r++ )
{
    for( int c = 0; c < NCOL; c++ )
    {
        if( (c == 0) || (c == NCOL-1) || (r == NROW-1) )
            board[r][c] = 1;
        else
            board[r][c] = 0;
    }
}
```

7. Write two C++ programs to test your answers to questions 2 and 3, using the following arrays (make sure your programs test all rows and columns), and assume MAXCOLS equals 100:

(a) 1 2 (b) 1 2 3
 3 4 4 5 6
 5 6

```
#include <iostream>
using namespace std;

const int MAXCOLS = 100;

int rowSum( int row, int A[ ][ MAXCOLS ], int nc );
int colSum( int col, int A[ ][ MAXCOLS ], int nr );

int main()
{
    int a[3][ MAXCOLS ] = { {1,2},{3,4},{5,6} };
    for( int r = 0; r < 3; r++ )
        cout << rowSum( r, a, 2 ) << endl;

    cout << endl;

    for( int c = 0; c < 2; c++ )
        cout << colSum( c, a, 3 ) << endl;
    cout << endl;

    int b[2][ MAXCOLS ] = { {1,2,3},{4,5,6} };
    for( r = 0; r < 2; r++ )
        cout << rowSum( r, b, 3 ) << endl;

    cout << endl << endl;

    for( c = 0; c < 3; c++ )
        cout << colSum( c, b, 2 ) << endl;
    cout << endl;

    return 0;
}

int rowSum( int row, int A[ ][ MAXCOLS ], int nc )
{
    int sum = 0;
    for( int c = 0; c < nc; c++ )
        sum = sum + A[row][c];
    return sum;
}

int colSum( int col, int A[ ][ MAXCOLS ], int nr )
{
    int sum = 0;
    for( int r = 0; r < nr; r++ )
        sum = sum + A[r][col];
    return sum;
}
```

8. Write a C++ program that tests your answer to question 4 for the following 3×3 arrays:

$$(a) \begin{matrix} 1 & 1 & 1 \\ -5 & 2 & -5 \\ -5 & 2 & -5 \end{matrix}$$

$$(b) \begin{matrix} 1 & 2 & -5 \\ 1 & 2 & 1 \\ 2 & 2 & 1 \end{matrix}$$

$$(c) \begin{matrix} 2 & 2 & 1 \\ 2 & 1 & -5 \\ 1 & -5 & 1 \end{matrix}$$

```
#include <iostream>
using namespace std;

const int MAXCOLS = 3;

int rowSum( int row, int A[ ][ MAXCOLS ], int nc );
int colSum( int col, int A[ ][ MAXCOLS ], int nr );
bool winner( int A[3][3] );

int main()
{
    int a[3][ MAXCOLS ] = { {1, 1, 1}, {-5, 2,-5}, {-5, 2,-5} };
    int b[3][ MAXCOLS ] = { {1, 2,-5}, { 1, 2, 1}, { 2, 2, 1} };
    int c[3][ MAXCOLS ] = { {2, 2, 1}, { 2, 1,-5}, { 1,-5, 1} };

    cout << boolalpha << winner( a ) << endl;
    cout << boolalpha << winner( b ) << endl;
    cout << boolalpha << winner( c ) << endl;

    return 0;
}

bool winner( int A[3][3] )
{
    for( int i = 0; i < 3; i++ )
    {
        int rsum = rowSum( i, A, 3 );
        int csum = colSum( i, A, 3 );
        if( rsum == 3 || csum == 3 || rsum == 6 || csum == 6 ) return true;
    }
    return false;
}

int rowSum( int row, int A[ ][ MAXCOLS ], int nc )
{
    int sum = 0;
    for( int c = 0; c < nc; c++ )
        sum = sum + A[row][c];
    return sum;
}

int colSum( int col, int A[ ][ MAXCOLS ], int nr )
{
    int sum = 0;
    for( int r = 0; r < nr; r++ )
        sum = sum + A[r][col];
    return sum;
}
```

Turn in, stapled together:

- (1) This sheet with your answers to questions 1-6 filled in.
- (2) Hardcopy listing of your answer to questions 7 and 8.