```
/*
   A mex function interface for the "GetTiles" routine

   by John L. Weatherwax
   email: wax@alum.mit.edu
*/

#include <iostream>
#include "tiles.h"
#include "stdlib.h"
#include "math.h"
#include "mex.h"

void
mexFunction( int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[] )
{
  int NUM_TILINGS, num_stat_vars, m, n, N, a;
  double *state_vars, *tiles_ptr;
  int *tiles_int_ptr, ti;
  mxArray *tiles;

    /* Check for proper number of input and output arguments
       could add more arguments to active the optional arguments ... which are now
deactivated */
    if ( nrhs != 4 ) {
        mexErrMsgTxt("Exactly four input arguments required.\n");
    }
    if( nlhs > 1 ){
        mexErrMsgTxt("Too many output arguments.\n");
    }

    /* extract the number of tilings */
    if( mxGetN(prhs[0]) != 1 || mxGetM(prhs[0]) !=1 ){
      mexErrMsgTxt("the first input argument must be a scalar integer.\n");
    }
    NUM_TILINGS = (int) mxGetScalar(prhs[0]);

    /* extract the state variables we will generate tiles for (and its dimension) */
    state_vars = mxGetPr(prhs[1]);
    m = mxGetM(prhs[1]);
    n = mxGetN(prhs[1]);
    num_stat_vars = 0;
    if( (m==1) && (n>=1) ){
      num_stat_vars = n;
    }
    if( (m>=1) && (n==1) ){
      num_stat_vars = m;
    }
    if( num_stat_vars==0 ){ /* the input dimensions are not correct */
      mexErrMsgTxt("the second input argument should be a [n,1] or [1,n] vector");
    }

    /* extract the number of possible hash parameters
     */
    if( mxGetN(prhs[2]) != 1 || mxGetM(prhs[2]) !=1 ){
      mexErrMsgTxt("the third input argument be a scalar integer.\n");
    }
    N = (int) mxGetScalar(prhs[2]);
```

```
    /* extract the "page" number of possible hash parameters
     */
    if( mxGetN(prhs[3]) != 1 || mxGetM(prhs[3]) !=1 ){
      mexErrMsgTxt("the fourth input argument be a scalar integer.\n");
    }
    a = (int) mxGetScalar(prhs[3]); a -= 1; /* convert input "a" to zero based indexing
*/

    /* allocate integer space for the ouput:
     */
    tiles_int_ptr = (int *) malloc( NUM_TILINGS*sizeof(int) );

    /* make the call the function that actually does the work
     */
    GetTiles(tiles_int_ptr,NUM_TILINGS,state_vars,num_stat_vars,N,a,-1,-1);

    /*tiles_ptr[0] = 1.0;
    tiles_ptr[1] = 2.0;
    tiles_ptr[2] = 3.0; */

    /* copy from the integer tiles to the double tiles
       -- and convert to ones based indexing (used by Matlab) by adding an additional ONE
     */
    tiles     = mxCreateDoubleMatrix(1,NUM_TILINGS,mxREAL);
    tiles_ptr = mxGetPr(tiles);
    for( ti=0; ti<NUM_TILINGS; ti++) tiles_ptr[ti] = (double) (tiles_int_ptr[ti]+1);

    free(tiles_int_ptr);

    plhs[0] = tiles; /* assign our work to the right hand side */

    return;
}
```