#include <iostream>

using namespace std;

struct matrix {

int rows, cols;

double \* arrayptr;

void print() {

cout << "rows = " << rows << " cols = " << cols << "\n";

}

matrix(int r, int c) {

rows = r;

cols = c;

arrayptr = new double[rows\*cols];

}

double get(int a, int b);

void set(int row, int col, double value);

void resize(int rsize, int csize);

matrix clone();

};

void matrix::resize(int rsize, int csize) {

cout << "resizing...\n";

int rowlim, collim;

if (rsize<0 || csize<0) {

cerr << "Error in resize\n";

return;

}

if (rsize<rows || csize<cols) {

cout << "Warning: New matrix size is smaller than current matrix. Continue? (Y or N)\n";

char answer;

cin >> answer;

if (answer == 'N') return;

}

else {

if (rsize>rows) rowlim = rows;

if (rsize<rows || rsize==rows) rowlim = rsize;

if (csize>cols) collim = cols;

if (csize<cols || csize==cols) collim = csize;

double \* newarray;

newarray = new double[rsize\*csize];

for (int i=0; i<rowlim; i++) {

for (int j=0; j<collim; j++)

newarray[i\*csize+j]=arrayptr[i\*cols+j];

}

delete [] arrayptr;

arrayptr = newarray;

rows = rsize;

cols = csize;

cout << "all done\n";

}

}

void matrix::set(int row, int col, double value) {

if (row<0 || col<0) {

cerr << "Error in set\n";

return;

}

if (row>rows && col>cols) resize(row+1,col+1);

else if (row>rows) resize(row+1,cols);

else if (col>cols) resize(rows,col+1);

arrayptr[row\*cols+col] = value;

}

double matrix::get(int a, int b) {

if (a<0 || a>=rows || b<0 || b>=cols) {

cerr << "Error in get\n";

return NULL;

}

return arrayptr[a\*cols+b];

}

matrix matrix::clone() {

matrix arrayclone(rows,cols);

arrayclone.arrayptr = new double[rows\*cols];

for (int i=0; i<rows\*cols; i++) {

arrayclone.arrayptr[i] = arrayptr[i];

}

return arrayclone;

}

void main() {

matrix m(20,20);

m.print();

m.set(10,10,123);

cout << "Value at [10][10] = " << m.get(10,10) << "\n";

m.set(30,30,22222);

matrix n = m.clone();

cout << "Value at [30][30] in m = " << m.get(30,30) << "\n";

cout << "Value at [30][30] in n = " << n.get(30,30) << "\n";

}

**Output:**

