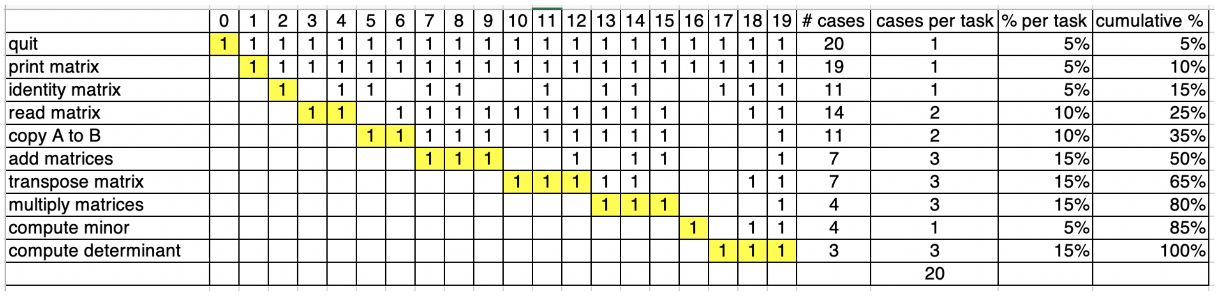
Computation I 5EIA2
C Exam: MatLab (v1.1, September 29, 2019)
30 September 17:30-20:30/21:00

In this exam you will develop a competitor to the Matlab program! The user will be able to create matrices, add, transpose, and multiply them, and compute the determinant.

Important

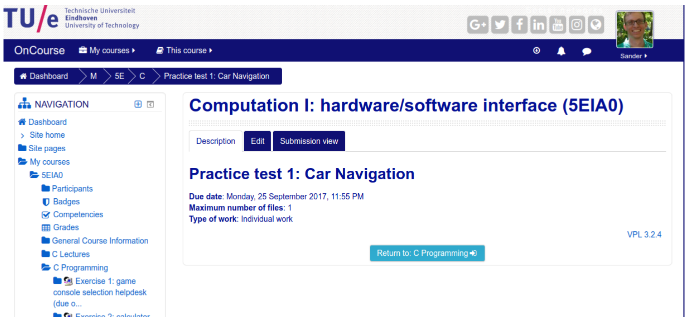
- Your grade is based on the number of cases passed. So try to complete the cases one by one. Note that there may be additional test cases that are only run after the exam.
- You can press "evaluate" as often as you like during the exam to evaluate your solution. We advise you to do this regularly during the exam.
- You must follow the instructions of the exam. For example, you may not use an array if a linked-list implementation is asked for.
- You are only allowed to use the Kernighan & Ritchie paper book in the Dutch or English language as a reference during the exam. No other electronic or printed material are allowed.
- The grade is based on your **last submission**. So make sure you submit a working version that completes as many cases as possible.
- Your program must work for all inputs, not just the test cases. We will change the test cases after the exam (but the number of test cases per task, i.e., the grade will not change).



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	# cases	cases per task	% per task	cumulative %
quit	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	20	1	5%	5%
print matrix		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19	1	5%	10%
identity matrix			1		1	1		1	1			1		1	1			1	1	1	11	1	5%	15%
read matrix				1	1		1	1	1	1	1	1	1	1	1	1			1	1	14	2	10%	25%
copy A to B						1	1	1	1	1		1	1	1	1	1				1	11	2	10%	35%
add matrices								1	1	1			1		1	1				1	7	3	15%	50%
transpose matrix											1	1	1	1	1				1	1	7	3	15%	65%
multiply matrices														1	1	1				1	4	3	15%	80%
compute minor																	1		1	1	4	1	5%	85%
compute determinant																		1	1	1	3	3	15%	100%
																						20		

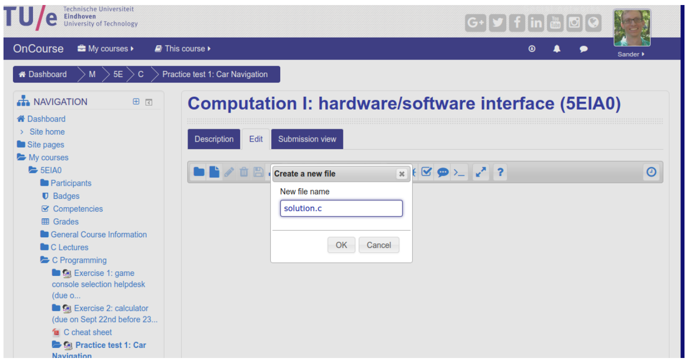
Figure 1: Test cases and points per task.

Open the correct C programming exam on exam.oncourse.tue.nl. You will see a screen similar to:


OnCourse
My courses
This course
Dashboard
Practice test 1: Car Navigation
NAVIGATION
Computation I: hardware/software interface (5EIA0)
Description
Edit
Submission view
Practice test 1: Car Navigation
Return to: C Programming

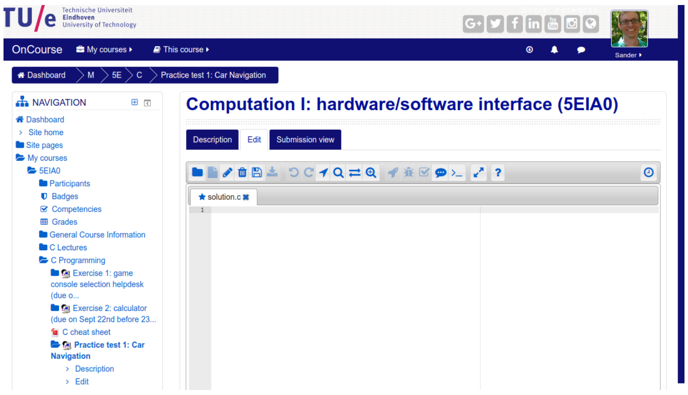
Due date: Monday, 25 September 2017, 11:55 PM
Maximum number of files: 1
Type of work: Individual work

Select the "Edit" tab and the following screen will appear:



New file name: solution.c

Provide a useful name for your C file (e.g., solution.c) and press "Ok".
You will then see the following screen:



You can write your own C program in the text editor that is now shown in your browser. Once you press "Save" you can "Run" and "Evaluate" your program. Using the "Run" command you will see a terminal where you can provide input to your program. You can use this to debug your program. Using the "Evaluate" command all test cases are evaluated and at the end the results are displayed (grade and errors if present).

Task 1. Write a C program that asks the user to select the command that needs to be performed. The commands that need to be supported are listed in the following table:

command	operation
q	quit program
I	create matrix A equal to the identity matrix
a	print matrix A
b	print matrix B
c	print matrix C
A	create matrix A
B	copy matrix A to matrix B
+	add matrix A to matrix B, placing the result in matrix C
t	transpose matrix A
*	multiply matrix A and B, placing the result in matrix C
m	compute a minor of matrix A, placing the result in matrix C
d	compute the determinant of matrix A

In this task you only need to implement the quit command. In later tasks you will implement the remainder.
Hint: make a loop in which you ask for a single character command, like this:

```
  char cmd;
  do {
    printf ("Command [qIabcAB+t*md]? ");
    scanf(" %c",&cmd); // notice the space before the %
    ...
  } while (cmd != 'q');
```

If an invalid character is given then print the error message `Invalid command 'Z'` (with Z replaced by the unknown command):

```
Command [qIabcAB+t*md]? X
Invalid command 'X'
Command [qIabcAB+t*md]? q
Bye!
```

Your program must produce the *exact* output, including all spaces, capitalisation, quotes, etc.
Your program should now pass test case 0.

Task 2. Define three two-dimensional matrices (`matrixA`, `matrixB`, `matrixC`) in your main function of `MAXSIZE` by `MAXSIZE` `float` elements and initialise them to zero. You also need to define the integer variables `rowsA`, `columnsA`, `rowsB`, `columnsB`, `rowsC`, `columnsC` that indicate the size of each matrix. They are initialised to 1.
Write a `print` function that prints a `matrix` of `rows` rows and `columns` columns. It prints the floating point numbers of each matrix row on one line, separated by a single space. The formatting of the `float` is a total of 6 positions with an accuracy of 3 positions. The `name` character is the name of the array. Use the letter 'a' for the command to print `matrixA` using this function.

```
#define MAXSIZE 10
void print (float matrix[MAXSIZE][MAXSIZE], int rows, int columns, char name) { ... }
int main (void) {
    ... print (matrixA,rowsA,columsA,'A'); ...
}
```

A 3x3 matrix would be formatted like this:

```
Command [qIabcAB+t*md]? a
Matrix A (3 X 3):
  1.000   0.000   0.000
  0.000   1.000   0.000
  0.000   0.000   1.000
```

Since the matrix `matrixA` has been initialised to a size of 1x1 with value 0 this should be the result of your program:

```
Command [qIabcAB+t*md]? a
Matrix A (1 X 1):
  0.000
Command? q
Bye!
```

Hint: Have a look in the Kernighan & Ritchie book in Appendix B1.3 to see how to format the floating point numbers (field width 6, accuracy 3).

Your program should now pass test cases 0 to 1.

Task 3. Next, in your main function add code to set `matrixA` to an identity matrix. Use the letter 'I' (captical letter 'i') for this command. An identity matrix is a square matrix (the number of rows equals the number of columns) in which all elements are zero, except those on the (i,i) diagonal, which are 1. Print an error message if the rows and columns are not the same, or if they are out of range.

```
Command [qIabcAB+t*md]? I
Size of matrix A (rows columns)? 4 4
Command [qIabcAB+t*md]? a
Matrix A (4 X 4):
  1.000   0.000   0.000   0.000
  0.000   1.000   0.000   0.000
  0.000   0.000   1.000   0.000
  0.000   0.000   0.000   1.000
Command [qIabcAB+t*md]? I
Size of matrix A (rows columns)? 11 11
Rows & colums must be equal and between 1 and 10
Command [qIabcAB+t*md]? I
Size of matrix A (rows columns)? 2 4
Rows & colums must be equal and between 1 and 10
Command [qIabcAB+t*md]? q
Bye!
```

Your program should now pass test cases 0 to 2.

Task 4. In this task you should write code in your main function to ask for the size of matrix `matrixA` and initialise `matrixA` with the floating point values entered by the user. Give an error message if the size of the matrix is out of range. Use the letter 'A' for this command. (Hint: use `%2d` in the row prompt.)

```
Command [qIabcAB+t*md]? a
Matrix A (1 X 1):
  0.000
Command [qIabcAB+t*md]? A
Size of matrix A (rows columns)? 1 11
Rows & colums must be between 1 and 10
Command [qIabcAB+t*md]? A
Size of matrix A (rows columns)? 2 6
Row  0? 1 2 3 4 5 6
Row  1? -1 -2 -3 -4 -5 -6
Command [qIabcAB+t*md]? a
Matrix A (2 X 6):
  1.000   2.000   3.000   4.000   5.000   6.000
 -1.000  -2.000  -3.000  -4.000  -5.000  -6.000
Command [qIabcAB+t*md]? q
Bye!
```

Your program should now pass test cases 0 to 4.

Task 5. Add the 'b' command to print `matrixB` and the 'c' command to print `matrixC`, by calling the `print` function that you wrote in Task 2. Next add the 'B' command that that copies `matrixA` to `matrixB`.

```
Command [qIabcAB+t*md]? I
Size of matrix A (rows columns)? 3 3
Command [qIabcAB+t*md]? a
Matrix A (3 X 3):
  1.000   0.000   0.000
  0.000   1.000   0.000
  0.000   0.000   1.000
Command [qIabcAB+t*md]? b
Matrix B (1 X 1):
  0.000
Command [qIabcAB+t*md]? c
Matrix C (1 X 1):
  0.000
Command [qIabcAB+t*md]? B
Command [qIabcAB+t*md]? b
Matrix B (3 X 3):
  1.000   0.000   0.000
  0.000   1.000   0.000
  0.000   0.000   1.000
Command [qIabcAB+t*md]? A
Size of matrix A (rows columns)? 1 3
Row  0? 10 11 12
Command [qIabcAB+t*md]? a
Matrix A (1 X 3):
 10.000  11.000  12.000
Command [qIabcAB+t*md]? b
Matrix B (3 X 3):
  1.000   0.000   0.000
  0.000   1.000   0.000
  0.000   0.000   1.000
Command [qIabcAB+t*md]? q
Bye!
```

Your program should now pass test cases 0 to 6.

Task 6. Add the add function to implement the '+' command to add `matrixA` to `matrixB`, placing the result in `matrixC`. If the dimensions of `matrixA` and `matrixB` are not the same then the error message `Dimensions of A & B do not match` should be printed and `matrixC` is left unchanged.

```
void add (
    float matrixA[MAXSIZE][MAXSIZE], int rowsA, int columnsA,
    float matrixB[MAXSIZE][MAXSIZE], int rowsB, int columnsB,
    float matrixC[MAXSIZE][MAXSIZE], int rowsC, int columnsC) { ...
```

Hint: Set `rowsC` and `columnsC` to the appropriate values *before* calling add.

Example output:

```
Command [qIabcAB+t*md]? A
Size of matrix A (rows columns)? 3 3
Row  0? 1 2 3
Row  1? 4 5 6
Row  2? 7 8 9
Command [qIabcAB+t*md]? B
Command [qIabcAB+t*md]? I
Size of matrix A (rows columns)? 3 3
Command [qIabcAB+t*md]? a
Matrix A (3 X 3):
  1.000   0.000   0.000
  0.000   1.000   0.000
  0.000   0.000   1.000
Command [qIabcAB+t*md]? b
Matrix B (3 X 3):
  1.000   2.000   3.000
  4.000   5.000   6.000
  7.000   8.000   9.000
Command [qIabcAB+t*md]? +
Command [qIabcAB+t*md]? c
Matrix C (3 X 3):
  2.000   2.000   3.000
  4.000   6.000   6.000
  7.000   8.000  10.000
Command [qIabcAB+t*md]? I
Size of matrix A (rows columns)? 2 2
Command [qIabcAB+t*md]? +
Dimensions of A & B do not match
Command [qIabcAB+t*md]? q
Bye!
```

Your program should now pass test cases 0 to 9.

Task 7. Implement the 't' command to transpose `matrixA`. To transpose a matrix each (i,j)th element is swapped with the (j,i)th element. Transposing a N*M matrix results in a M*N matrix.

```
Command [qIabcAB+t*md]? A
Size of matrix A (rows columns)? 2 3
Row  0? 1 2 3
Row  1? 4 5 6
Command [qIabcAB+t*md]? a
Matrix A (2 X 3):
  1.000   2.000   3.000
  4.000   5.000   6.000
Command [qIabcAB+t*md]? t
Command [qIabcAB+t*md]? a
Matrix A (3 X 2):
  1.000   4.000
  2.000   5.000
  3.000   6.000
Command [qIabcAB+t*md]? q
Bye!
```

Your program should now pass test cases 0 to 12.

Task 8. Add the `mult` function to implement the '*' command to multiply `matrixA` with `matrixB`, placing the result in `matrixC`. If the dimensions of `matrixA` and `matrixB` do not match then the error message `Dimensions of A & B do not match` should be printed and `matrixC` is left unchanged.

```
 void add (
     float matrixA[MAXSIZE][MAXSIZE], int rowsA, int columnsA,
     float matrixB[MAXSIZE][MAXSIZE], int rowsB, int columnsB,
     float matrixC[MAXSIZE][MAXSIZE], int rowsC, int columnsC) { ...
```

Recall that the formula to multiply two matrices A (rowsA X columnsA) and B (rowsB X columnsB) to result in C (rowsA X columnsB) is as follows:

$$\forall 0 \le r < \texttt{rowsA}.\ \ \forall 0 \le c < \texttt{columnsB}.\ \ C(r,c) = \sum_{0 \le i < \texttt{columnsA}} A(r,i) \times B(i,c)$$

(The $\forall$ symbol means "for all". The $\sum$ formula means adding `A(r,0)`$\times$ `B(0,c) + A(r,1)`$\times$ `B(1,c) + ... + A(r,columnsA-1)`$\times$ `B(columnsA-1,c)`.)

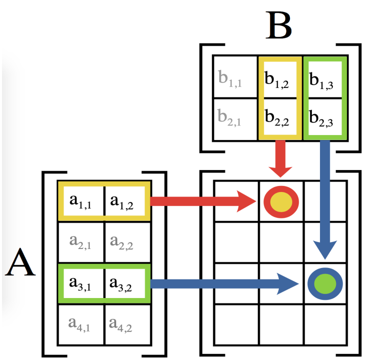


Figure 2: Matrix C is the multiplication of matrix A with matrix B.

For example, the multiplication of 1 X 5 and 5 X 1 matrices results in a 1 X 1 matrix, and the multiplication of 5 X 1 and 1 X 5 matrices results in a 5 X 5 matrix.

```
Command [qIabcAB+t*md]? A
Size of matrix A (rows columns)? 2 3
Row  0? 1 2 3
Row  1? 4 5 6
Command [qIabcAB+t*md]? B
Command [qIabcAB+t*md]? t
Command [qIabcAB+t*md]? a
Matrix A (3 X 2):
  1.000   4.000
  2.000   5.000
  3.000   6.000
Command [qIabcAB+t*md]? b
Matrix B (2 X 3):
  1.000   2.000   3.000
  4.000   5.000   6.000
Command [qIabcAB+t*md]? *
Command [qIabcAB+t*md]? c
Matrix C (3 X 3):
 17.000  22.000  27.000
 22.000  29.000  36.000
 27.000  36.000  45.000
Command [qIabcAB+t*md]? q
Bye!
```

Your program should now pass test cases 0 to 15.

Task 9. Add the 'm' command to compute the minor of `matrixA` and place the result in `matrixC`. The minor `minor(A,i,j)` of matrix A is matrix A with row i and column j removed. In the figure below, the minors `minor(A,0,0)`, `minor(A,0,1)`, and `minor(A,0,2)` have been highlighted.

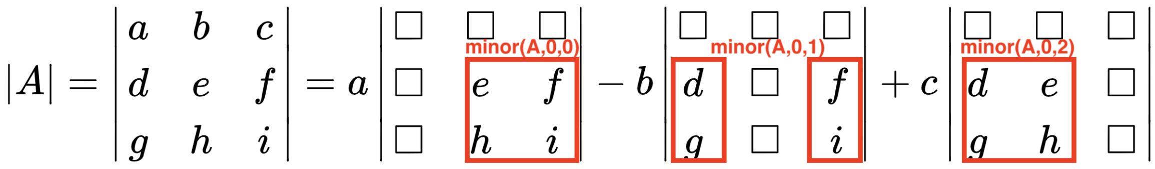


Figure 3: Minors (0,0), (0,1), and (0,2) of a 3 X 3 matrix

Write a function `minormatrix`:

```
void minormatrix (float matrixA[MAXSIZE][MAXSIZE], int rowsA, int columnsA,
                  int r, int c, float min[MAXSIZE][MAXSIZE]) ...
int main(void) {
  ... minormatrix(matrixA,rowsA,columsA,r,c,matrixC); ...
```

that removes row `r` and column `c` from `matrixA` and places the result in matrix `min`. (Set `rowsC` and `columnsC` to the right size before calling `minormatrix`.)
If `matrixA` has fewer than 2 rows or columns or if the specified row or column is out of range then give an error message, as shown below.

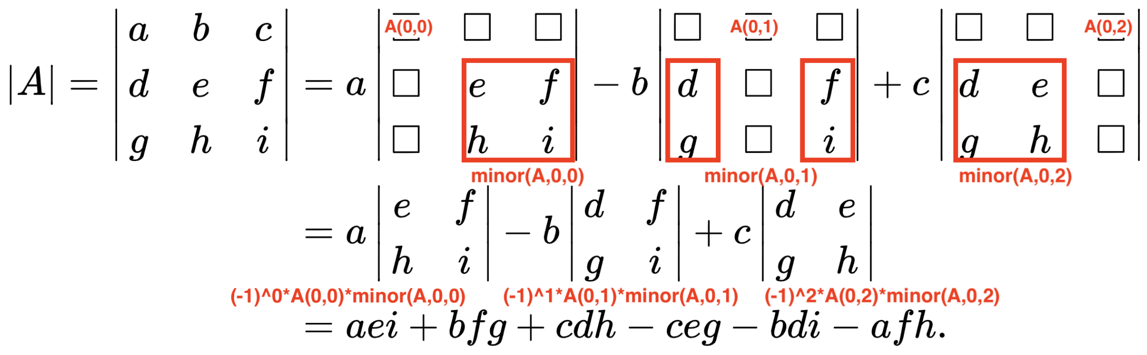
```
Command [qIabcAB+t*md]? a
Matrix A (4 X 3):
  3.000   2.000   1.000
 -3.000  -5.000  -7.000
 10.000  20.000  30.000
-10.000 -11.000 -12.000
Command [qIabcAB+t*md]? m
Remove which row & column of matrix A? 0 0
Command [qIabcAB+t*md]? c
Matrix C (3 X 2):
 -5.000  -7.000
 20.000  30.000
-11.000 -12.000
Command [qIabcAB+t*md]? m
Remove which row & column of matrix A? 1 2
Command [qIabcAB+t*md]? c
Matrix C (3 X 2):
  3.000   2.000
 10.000  20.000
-10.000 -11.000
Command [qIabcAB+t*md]? m
Remove which row & column of matrix A? 0 5
Rows & colums must be between 0 and 3 & 2, respectively
Command [qIabcAB+t*md]? I
Size of matrix A (rows columns)? 1 1
Command [qIabcAB+t*md]? m
Remove which row & column of matrix A? 0 0
Matrix A must have at least two rows & colums
Command [qIabcAB+t*md]? q
Bye!
```

Your program should now pass test cases 0 to 16.

Task 10. The final task is to compute the determinant of matrixA using the Laplace expansion. Don't worry, this is what you learned in high school. This is a recursive way of computing a determinant that uses the `minor` function of the previous task.
The determinant of a 1 X 1 matrix A is equal to A(0,0). The determinant of a 2 X 2 matrix A is equal to A(0,0)*A(1,1) - A(0,1)*A(1,0). The determinant of a 3 X 3 matrix A is computed by taking row zero and multiplying the elements (r,c) of the row by the minors `minor(A,r,c)`, as illustrated in the figure below. The only minor complication is that the sign of each product is either $+1$ or -1, as defined by $(-1)^c$. (You can use any row or column, but we recommend using row 0, as shown in the formula and figure.)

$$\texttt{determinant}(A) = \sum_{0 \le c < \texttt{columnsA}} (-1)^c \times \texttt{determinant}(\texttt{minor(A,0,c)})$$

The formula above allows you to compute the determinant of any N X N matrix.


A(0,0)
A(0,1)
A(0,2)
minor(A,0,0)
minor(A,0,1)
minor(A,0,2)
(-1)^0*A(0,0)*minor(A,0,0)
(-1)^1*A(0,1)*minor(A,0,1)
(-1)^2*A(0,2)*minor(A,0,2)

$$|A| = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix} = aei + bfg + cdh - ceg - bdi - afh.$$

Figure 4: The minors of a 3 X 3 matrix

You can use the following code as a template.

```
float determinant (float matrixA[MAXSIZE][MAXSIZE], int rowsA, int columnsA)
{
  float det;
  if (rowsA == 1 && columnsA == 1) det = ...;
  else if (rowsA == 2 && columnsA == 2) det = ...;
  else {
    // recursively call determinant on the columnsA minors of A along row 0
    int sign = (c % 2 ? -1 : +1); // the sign for minor(matrixA,0,c)
  }
  printf("%d",rowsA);
  return det;
}
```

If `matrixA` is not square then then give the error message `Matrix A must be square`
The `printf` in the template code is required and it shows the recursion, e.g. 2223 for a 3 X 3 matrix shows that determinant has been called three times on a 2 X 2 matrix, and once on a 3 X 3 matrix. For a 4 X 4 matrix the recursion becomes clear as determinant is called four times on a 3 X 3 matrix: 22232223222322234.

```
Command [qIabcAB+t*md]? I
Size of matrix A (rows columns)? 3 3
Command [qIabcAB+t*md]? d
2223
The determinant is 1.000000
Command [qIabcAB+t*md]? I
Size of matrix A (rows columns)? 4 4
Command [qIabcAB+t*md]? d
22232223222322234
The determinant is 1.000000
Command [qIabcAB+t*md]?
```

Your code should now pass all test cases.

```
Command [qIabcAB+t*md]? A
Size of matrix A (rows columns)? 1 1
Row  0? 9
Command [qIabcAB+t*md]? d
1
The determinant is 9.000000
Command [qIabcAB+t*md]? A
Size of matrix A (rows columns)? 2 2
Row  0? 1 2
Row  1? 10 11
Command [qIabcAB+t*md]? d
2
The determinant is -9.000000
Command [qIabcAB+t*md]? A
Size of matrix A (rows columns)?
3  3
Row  0? -2 2 -3
Row  1? -1 1 3
Row  2? 2 0 -1
Command [qIabcAB+t*md]? d
2223
The determinant is 18.000000
Command [qIabcAB+t*md]? A
Size of matrix A (rows columns)? 4 4
Row  0? 1 2 -4 -5
Row  1? 2 7 3 -2
Row  2? 1 1 -1 -1
Row  3? 4 7 8 -5
Command [qIabcAB+t*md]? d
22232223222322234
The determinant is -200.000000
```

Submission: Your *last* submission will be automatically graded.

- **29/9 v1.1** Minor clarifications.