Final project: Linear algebra calculator

General description

In this project you should write a linear algebra calculator. Your program should read from the user series of commands and then parse and execute them. The commands your program should support are all basic linear algebra operations and are listed below.

Your program should support definition of "variables" that can be of different types – matrices (square matrices only), vectors and scalars. Each variables will be identified by a name - capital case letter 'A'-'Z' (this means that there are at most 26 such "variables"). The variables types are marked with lower case letters: 'm' (for a matrix), 'v' (vector) and 's' (scalar). The matrices and vectors can come in different dimensions. If needed, you can assume the maximal dimension is 6 (i.e. the biggest matrices you will need to face is 6X6).

The program should start by asking the user to give an input source and reading the answer from the keyboard. The input source could be either a file (given by the filename) or the keyboard itself (given by either an empty string or "keyboard"). The program should then follow the commands given from the input source (one command per line), until the HALT command is given.

You may choose to work in pairs.

Commands

The calculator should except and execute the following commands:

1. SET <variable_name> <variable_type> <dimension> <comma separated values>:
   Should set <variable_name> to be a of type <variable_type> with dimension <dimension> and values given by the <comma separated values>. When setting a scalar, the <dimension> parameter has no meaning and should be ignored.
   The program should make sure that <variable_name> <variable_type> and <dimension> are valid and that the number of values given fit the dimension and type.

   Examples:
   The command "SET W m 3 1,2,3,4.7,5,6.8,7,8,9" should set $W = \begin{pmatrix} 1 & 2 & 3 \\ 4.7 & 5 & 6.8 \\ 7 & 8 & 9 \end{pmatrix}$

   The command "SET X v 5 10,20,30,40,55.8" should set $X = (10 \quad 20 \quad 30 \quad 40 \quad 55.8)$

   The command "SET A s 4 100" should set A=100

2. PRINT <variable_name>
   Should print to screen the name, type, dimension (if relevant) and value of <variable_name>.

3. ADD <variable_name1> <variable_name2> <variable_name3>
   Should set <variable_name3> to be the sum of <variable_name1> and <variable_name2>. Addition is allowed only between variables of the same type and dimension. Warning should be printed if this not the case.
   Examples:
   "ADD A B C" – should set C = A + B (assuming A and B have the same type and dimension)

4. MULTIPLY <variable_name1> <variable_name2> <variable_name3>
   Should set <variable_name3> to be the multiplication of <variable_name1> and <variable_name2>. Multiplication is allowed only between the following types:
   - Matrix and a matrix – resulting with a matrix
   - Matrix and a vector – resulting with a vector.
   - Vector and a vector – resulting with a scalar (dot product).
   - Scalar with any other type – resulting with the same type.

   Multiplication is only valid between "variables" with the same dimension (except when multiplying with a scalar).

5. OUTER_PRODUCT <variable_name1> <variable_name2> <variable_name3>
   Should set <variable_name3> to be the outer product of <variable_name1> and <variable_name2>. This operation is only defined between two vectors of the same dimension (resulting with a matrix). Reminder: the outer product $M = u \otimes u$, where u and v are vectors of the same dimension is a matrix M given by $M_{ij} = u_i v_j$.

6. HALT
   The program should terminate

7. HELP
   Should print to screen a short help message listing the allowed command and their syntax.

Warnings

Your program should print warning whenever an invalid command is given. Invalid command can be a command that does not follow the syntax of the program or cannot be executed.

Examples:

The command “SET a v 3 10,20,30” should print the warning “Variable name ‘a’ is illegal”

The command “SET C m 2 5,7,10,15,16” should print the warning “number of values given does not fit the input type and dimension”

The command “DIVIDE M V E” should print the warning “command DIVIDE is not defined”

The command “ADD A B C” should print the warning “variable name A is not defined” if variable A was not defined prior to this command.

NOTE: the list of warnings above does not cover the full list of invalid commands you may encounter. Part of your work is to make sure you cover all of the problematic input you may be given by the user.

Grading

50 points – Functionality (i.e. the program does what it should)

10 points – Documentation. Including:

- Brief description of the program at the top of the file.
- Short description of each function including its parameters and returned value (if relevant)
- Everywhere you think the code is not self-explanatory

10 points – following coding best practices, including: meaningful variables names, avoiding code duplication, splitting to functions, avoiding “magic numbers”.

20 points - Oral exam. Upon submitting your project, we will ask you few questions about your implementation. This part is individual even if you chose to work in pairs.

20 points – Memory efficiency. Write your program to use the lowest amount of memory possible. Note: this part might be hard to implement. If you find the project too hard, you can choose to make it easier by writing a memory inefficient program (your maximal grade will be 90 in this case). It is advised to decide on this manner before you start implementing.

Submission

When you submit your project, you will meet with one of us in a computer lab. You will be asked to compile your project (on visual studio) and ran the program with few input files to test its functionality. You will also be asked few question on your implementation (oral exam). In addition, you are asked to submit a hard copy of your code.

Due date is 26.07.2017. There will be an optional submission day before the exam (probably on 6.7). It is highly recommended that you will work on your project before the test.

General remarks

- **A well designed structure will make your life much easier.**
- The program is quite involved. Try to break it into relatively simple tasks and handle them separately. It is easier to find errors in small portion of the code dealing with a limited task. Thus, intelligent use of functions considerably facilitates the work.
- Use double precision variables whenever non-integer values are needed.
- When reading the commands, consecutive spaces should be treated as one.
- Test your program under various inputs.

IMPORTANT

You should do the project by yourself, do not share your code (other than with your pair).

We have opened a forum in Moodle for the project. Any question you might have about the project, should be asked via the forum were everyone can see the question and our answers. You are not allowed to post code in the forum.