The Programming Assignment Report Instructions

CSCE 221

1. The description of an assignment problem.

The assignment is to implement a 2d dynamic array with pointers to read, operate and output data in C++.

2. The description of data structures and algorithms used to solve the problem.

(a) Provide definitions of data structures by using Abstract Data Types (ADTs)

ADTs is an abstract model to specifies the operations that support the data.

(b) Write about the ADTs implementation in C++.

The public member functions of class in C++ correspond to the operations of the ADTs.

(c) Describe algorithms used to solve the problem.

Create an empty 2d dynamic array use pointers, set the elements from input to the array, do the operations and. output.

(d) Analyze the algorithms according to assignment requirements.

Use constructors to create 2d array. Read the inputs through input operator. Set the elements to array. Do the. multiplication and addtion operation. Output.

3. A C++ organization and implementation of the problem solution

(a) Provide a list and description of classes or interfaces used by a program such as classes used to implement the data structures or exceptions.

Defaut constructor: set variable to 0, and pointer to nullptr.

Paremeter constructor: set rows and columns, create a 2d dynamic array

Copy constructor/assignment operater: copy

Move constructor/assignment operater: copy and delete originals

Accessors: return elements in array

Mathmatical operator: do multiplication and addition

i/o operator: input and output elements

(b) Include in the report the class declarations from a header file (.h) and their implementation from a source file (.cpp).

```
/* My_matrix.h
```

```
Header file for the class My_matrix

Write your name and UIN here

*/

#include <iostream>
#include <exception>

using namespace std;
// Definitions of user defined type exceptions
struct invalid_input : public exception {
    virtual const char* what() const throw()
    { return "Invalid matrix input"; }
};

struct incompatible_matrices : public exception {
    virtual const char* what() const throw()
    { return "Incompatible matrices"; }
};

class My_matrix {

    //member variables
    int n, m;
    int **ptr;
    void allocate_memory();    // optional
public:
    //member functions
    My_matrix();    // default constructor
    My_matrix(int n1, int m1); // parameter constructor
    ~My_matrix(); // destructor
    My_matrix(const My_matrix& mat); // copy constructor
    My_matrix(My_matrix&& mat);    // move constructor (optional)
    My_matrix& operator=(const My_matrix& mat); //copy assignment operator
    My_matrix& operator=(My_matrix&& mat); // move assignment operator (optional)
    int number_of_rows() const;
    int number_of_columns() const;
    int* operator()(int i) const; // overloaded to access to ith row
    int& operator()(int i, int j); // overloaded to access (i,j) element
    int operator()(int i, int j) const; // overloaded to access (i,j) element
    int elem(int i, int j) const; // overloaded to access (i,j) element
    int& elem(int i, int j); // overloaded to access (i,j) element
};

// see the handout for the description of these operators
istream& operator>>(istream& in, My_matrix& mat);
ostream& operator<<(ostream& out, const My_matrix& mat);
My_matrix operator+(const My_matrix& mat1, const My_matrix& mat2);
My_matrix operator*(const My_matrix& mat1, const My_matrix& mat2);
```

```
/* My_matrix.h

Header file for the class My_matrix

Write your name and UIN here
```

```
*/

#include <iostream>
#include <exception>

using namespace std;
// Definitions of user defined type exceptions
struct invalid_input : public exception {
    virtual const char* what() const throw()
    { return "Invalid matrix input"; }
};

struct incompatible_matrices : public exception {
    virtual const char* what() const throw()
    { return "Incompatible matrices"; }
};

class My_matrix {

    //member variables
    int n, m;
    int **ptr;
    void allocate_memory();    // optional

public:
    //member functions
    My_matrix();    // default constructor
    My_matrix(int n1, int m1); // parameter constructor

    ~My_matrix(); // destructor

    My_matrix(const My_matrix& mat); // copy constructor

    My_matrix(My_matrix&& mat);    // move constructor (optional)

    My_matrix& operator=(const My_matrix& mat); //copy assignment operator

    My_matrix& operator=(My_matrix&& mat); // move assignment operator (optional)

    int number_of_rows() const;

    int number_of_columns() const;

    int* operator()(int i) const; // overloaded to access to ith row

    int& operator()(int i, int j); // overloaded to access (i,j) element

    int operator()(int i, int j) const; // overloaded to access (i,j) element

    int elem(int i, int j) const; // overloaded to access (i,j) element

    int& elem(int i, int j); // overloaded to access (i,j) element

};


// see the handout for the description of these operators

istream& operator>>(istream& in, My_matrix& mat);

ostream& operator<<(ostream& out, const My_matrix& mat);

My_matrix operator+(const My_matrix& mat1, const My_matrix& mat2);

My_matrix operator*(const My_matrix& mat1, const My_matrix& mat2);
```

(c) Provide features of the C++ programming paradigms like Inheritance or Polymorphism in case of object oriented programming, or Templates in the case of generic programming used in your implementation.

The template in my implementation can handle any numeric type.

4. A user guide description how to navigate your program with the instructions how to:

(a) compile the program: specify the directory and file names, etc.

Main.cpp, My_matrix.cpp, My_matrix.h

Put all the file in a folder and type the command respectively:

```
make all
```

(b) run the program: specify the name of an executable file.

```
Main.exe
```

5. Specifications and description of input and output formats and files

. (a) The type of files: keyboard, text files, etc (if applicable).

. txt file or user inputs.

. (b) A file input format: when a program requires a sequence of input items, specify the number of items per line or a line termination. Provide a sample of a required input format.

. Ex. Input.txt

```
. 3 3

. 1 2 3

. 1 2 3

. 1 2 3
```

. (c) Discuss possible cases when your program could crash because of incorrect input (a wrong file name, strings instead of a number, or such cases when the program expects 10 items to read and it finds only 9.)

. Invalid input like 2.0 when the variable type is int.

6. Provide types of exceptions and their purpose in your program.

7. (a) logical exceptions (such as deletion of an item from an empty container, etc.).

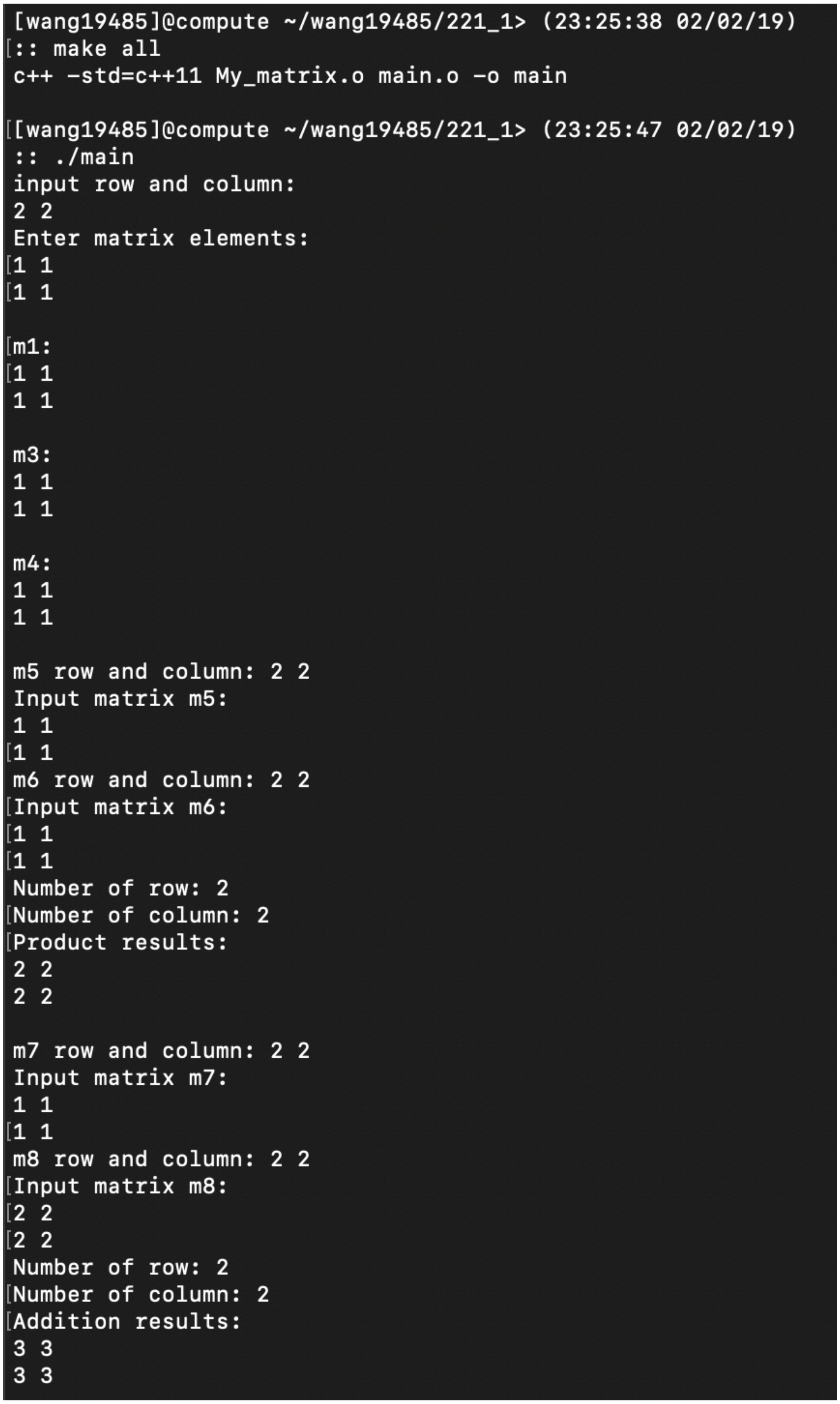
   If user input an invalid file, like missing an element, the exception will be throwed.

8. (b) runtime exception (such as division by 0, etc.)

   Out of range like i > n, j > m

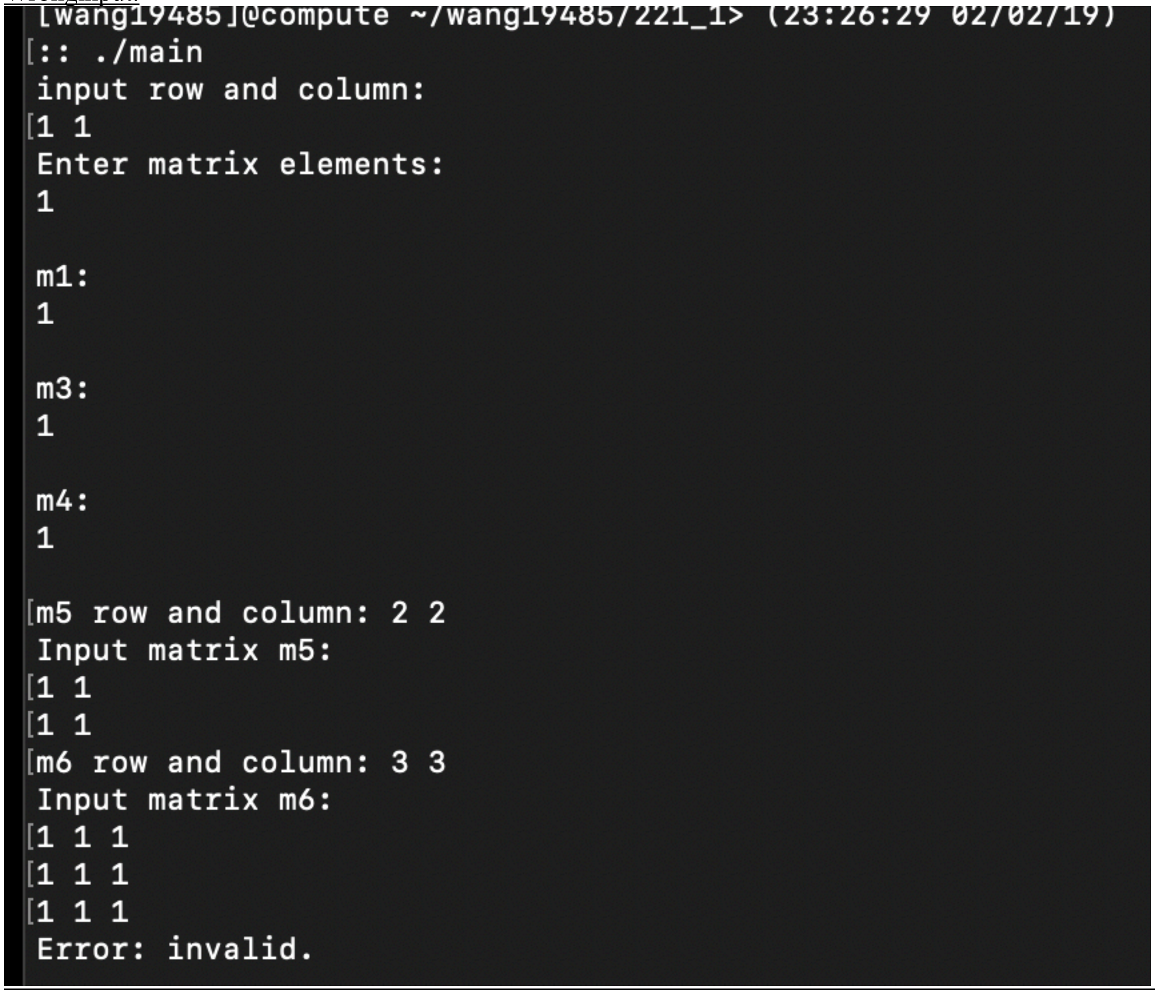
9. Test your program for correctness using valid, invalid, and random inputs (e.g., insertion of an item at the beginning, at the end, or at a random place into a sorted vector). Include evidence of your testing, such as an output file or screen shots with an input and the corresponding output.

   <u>Correct user input evidence:</u>



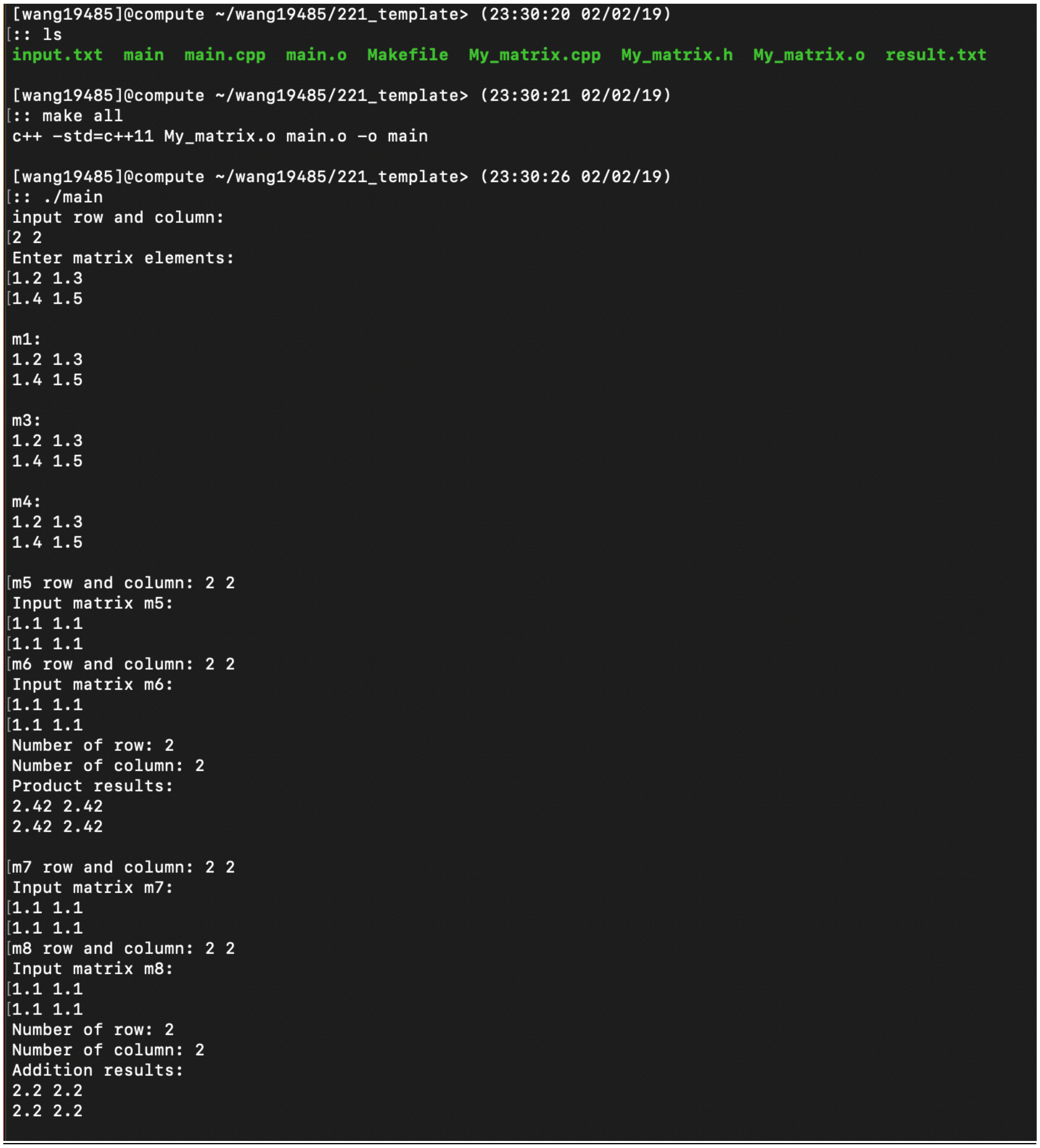
```
[wang19485]@compute ~/wang19485/221_1> (23:25:38 02/02/19)
[:: make all
c++ -std=c++11 My_matrix.o main.o -o main

[[wang19485]@compute ~/wang19485/221_1> (23:25:47 02/02/19)
:: ./main
input row and column:
2 2
Enter matrix elements:
[1 1
[1 1

[m1:
[1 1
1 1

m3:
1 1
1 1

m4:
1 1
1 1

m5 row and column: 2 2
Input matrix m5:
1 1
[1 1
m6 row and column: 2 2
[Input matrix m6:
[1 1
[1 1
Number of row: 2
[Number of column: 2
[Product results:
2 2
2 2

m7 row and column: 2 2
Input matrix m7:
1 1
[1 1
m8 row and column: 2 2
[Input matrix m8:
[2 2
[2 2
Number of row: 2
[Number of column: 2
[Addition results:
3 3
3 3
```

Wronginput:

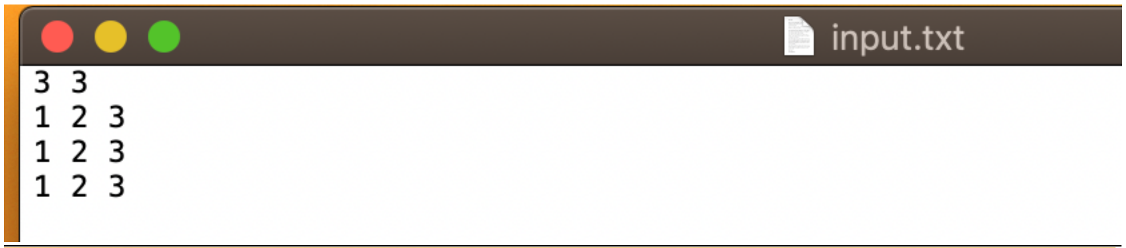


```
[wang19485]@compute ~/wang19485/221_1> (23:26:29 02/02/19)
[:: ./main
input row and column:
[1 1
Enter matrix elements:
1

m1:
1

m3:
1

m4:
1

[m5 row and column: 2 2
Input matrix m5:
[1 1
[1 1
[m6 row and column: 3 3
Input matrix m6:
[1 1 1
[1 1 1
[1 1 1
Error: invalid.
```

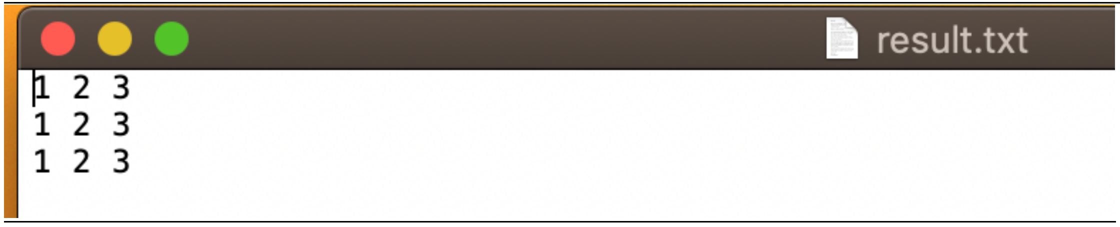
For template:



```
[wang19485]@compute ~/wang19485/221_template> (23:30:20 02/02/19)
[:: ls
input.txt  main  main.cpp  main.o  Makefile  My_matrix.cpp  My_matrix.h  My_matrix.o  result.txt

[wang19485]@compute ~/wang19485/221_template> (23:30:21 02/02/19)
[:: make all
c++ -std=c++11 My_matrix.o main.o -o main

[wang19485]@compute ~/wang19485/221_template> (23:30:26 02/02/19)
[:: ./main
input row and column:
[2 2
Enter matrix elements:
[1.2 1.3
[1.4 1.5

m1:
1.2 1.3
1.4 1.5

m3:
1.2 1.3
1.4 1.5

m4:
1.2 1.3
1.4 1.5

[m5 row and column: 2 2
Input matrix m5:
[1.1 1.1
[1.1 1.1
[m6 row and column: 2 2
Input matrix m6:
[1.1 1.1
[1.1 1.1
Number of row: 2
Number of column: 2
Product results:
2.42 2.42
2.42 2.42

[m7 row and column: 2 2
Input matrix m7:
[1.1 1.1
[1.1 1.1
[m8 row and column: 2 2
Input matrix m8:
[1.1 1.1
[1.1 1.1
Number of row: 2
Number of column: 2
Addition results:
2.2 2.2
2.2 2.2
```



```
3 3
1 2 3
1 2 3
1 2 3
```



```
1 2 3
1 2 3
1 2 3
```

Sample input and output file.