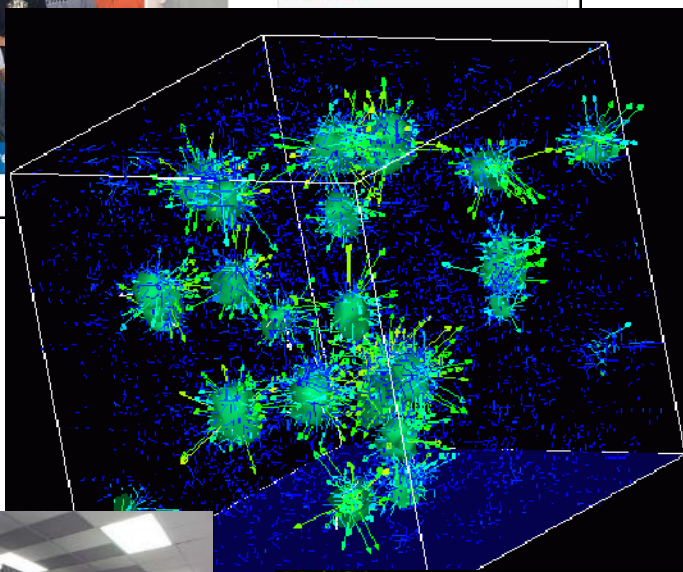


Survey of Scientific Computing (SciComp 301)

Dave Biersach
Brookhaven National
Laboratory
dbiersach@bnl.gov



```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Windows.Forms;
9
10 namespace SimpleEvents
11 {
12     public partial class Form1 : Form
13     {
14         Person person = new Person();
15
16         public Form1()
17         {
18             InitializeComponent();
19             person.FirstName = "Christian";
20             person.LastName = "Pano";
21         }
22
23         private void button1_Click(object sender, EventArgs e)
24         {
25             person.MainColor = textBox1.Text;
26         }
27     }
28 }
```

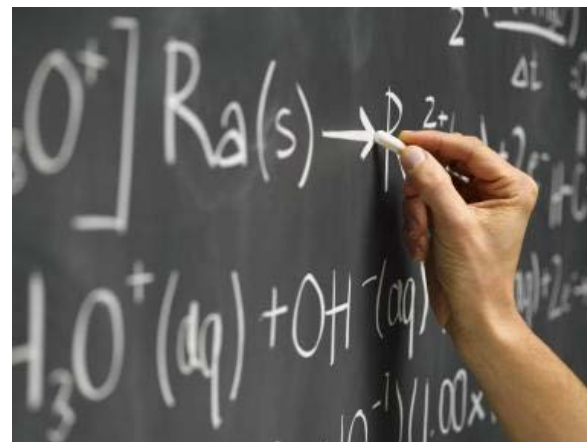
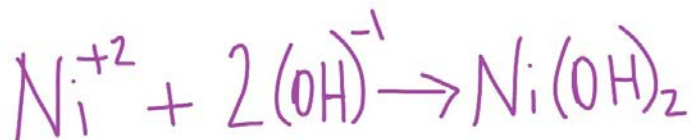
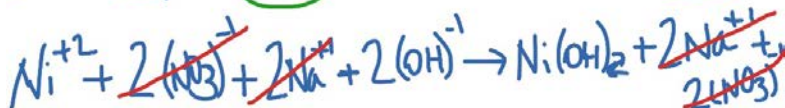
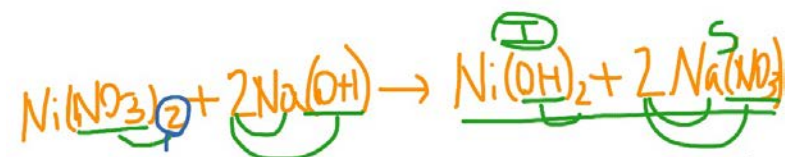
Session 17
Computational Chemistry,
Clustering

Session Goals

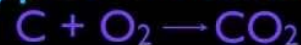
- Balance **ionic chemical equations** using linear programming
- Correlate “principle of atom conservation” (**POAC**) to the *minimization* of an objective function
- Represent molar & charge ratios as **linear constraints**
- Encode chemical equations as a matrix within a text file
- Implement **brute-force** searching using a **stack** data structure *instead* of using recursion or nested **for()** loops

Balancing Ionic Equations

- Can we write a program to balance ionic equations?

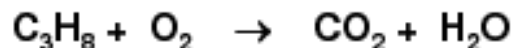


5 Types of Reactions

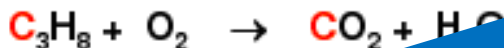


Balancing Ionic Equations

Step 1: Write reactants and products

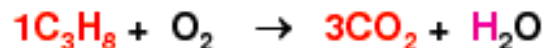


Step 2: Find one atom that occurs only in one substance on both sides



Step 3: Find coefficient

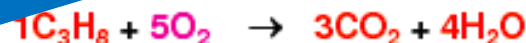
Step 4: Find another unbalanced atom which occurs only in one substance



Step 5: Find coefficient to balance this atom



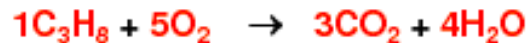
Step 6: Balance



Step 7: Make sure the coefficients are reduced to the smallest whole number values



Step 8: Check the balanced equation



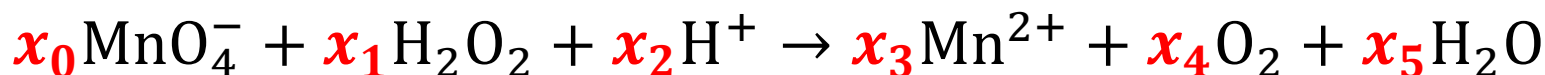
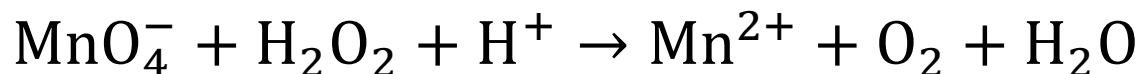
3C, 8H, 10O

3C, 10O, 8H

Yikes - and this is not even ionic!

Chemistry is Optimization

- Consider a chemical formula as an **optimization** problem:

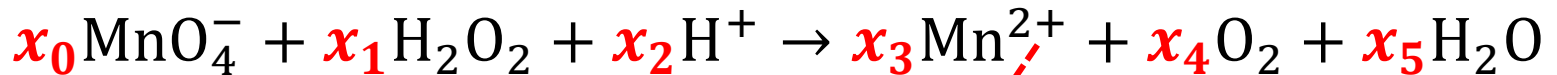


- Develop the **objective function** and the **constraints**:

$$\text{obj: minimize } \sum_{i=0}^n x_i$$

due to Principle of Atom Conservation (**POAC**)

Chemistry is Optimization



obj: minimize $\sum_{i=0}^n x_i$

$x_0 = x_3$	To balance the Manganese (Mn)
$4x_0 + 2x_1 = 2x_4 + x_5$	To balance the Oxygen (O)
$2x_1 + x_2 = 2x_5$	To balance the Hydrogen (H)
$-x_0 + x_2 = 2x_3$	To balance the charges

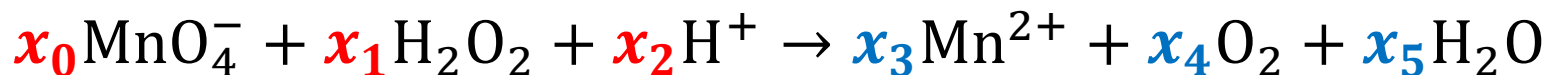
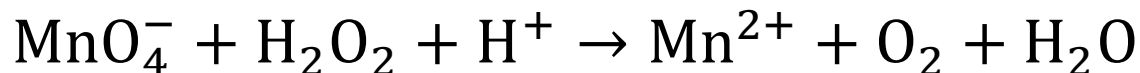
Chemistry is Optimization

- We have transformed solving a chemical equation into a solving a **linear optimization** problem
 - The **Simplex** method is a known good approach to solve linear programming problems in reasonable time
 - However managing **slack variables** and the **tableau** algorithm can be intimidating for novice programmers
- For small problems, we can just enumerate the ***search space*** (defined by all *permutations* of coefficients subject to the constraints) in order to find all of the possible solutions!

Encoding an Chemical Equation

- We can encode a chemical equation using a **2D matrix**
 - There will be a horizontal row for each **element** (atom)
 - We may need another row to encode the **ionic charges**
 - The **last row** will always contain the **maximum values** we want to consider for each unknown coefficient
- Each molecule is a term in the equation
 - There will be a vertical column for each **molecule** (or charged ion)
 - Every column is another **unknown** we need to resolve
- The matrix elements will be stored in a text (.txt) file
 - The first line will describe the **# of cols** and **# of rows**

Encoding Chemical Equation 1



One row for every atom and a row for charges *if needed*

Manganese (Mn)
Oxygen (O)
Hydrogen (H)
Charges (+/-)
Max Value

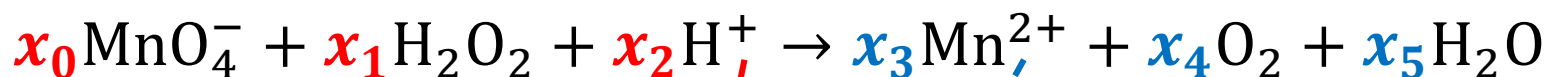
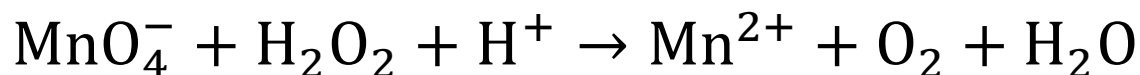
The matrix will have **5 rows** and **6 columns**

A column for every x_n

The last row is always the search limit for each x_n

File	Edit	View	Search	Tools	Documents	Help
5	6					
1	0	0	-1	0	0	
4	2	0	0	-2	-1	
0	2	1	0	0	-2	
-1	0	1	-2	0	0	
9	9	9	9	9	9	

Encoding Chemical Equation 1



Reactant values are
copied verbatim

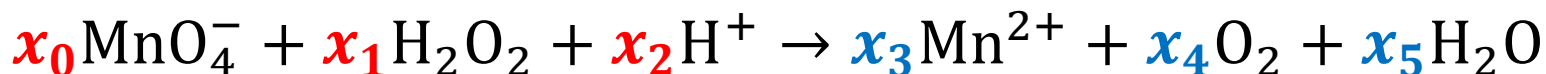
Manganese (Mn)
Oxygen (O)
Hydrogen (H)
Charges (+/-)
Max Value

Open ▾		+ eqn1.txt ~/Desktop/scic...		Save		- + ×	
File	Edit	View	Search	Tools	Documents	Help	
5	6						
1	0	0	-1	0	0		
4	2	0	0	-2	-1		
0	2	1	0	0	-2		
-1	0	1	-2	0	0		
9	9	9	9	9	9		

Product values are
negated !!

Encoding Chemical Equation 1

- A potential **valid** set of unknown values will yield a **sum of zero** when convolved with each row (constraint) in the matrix

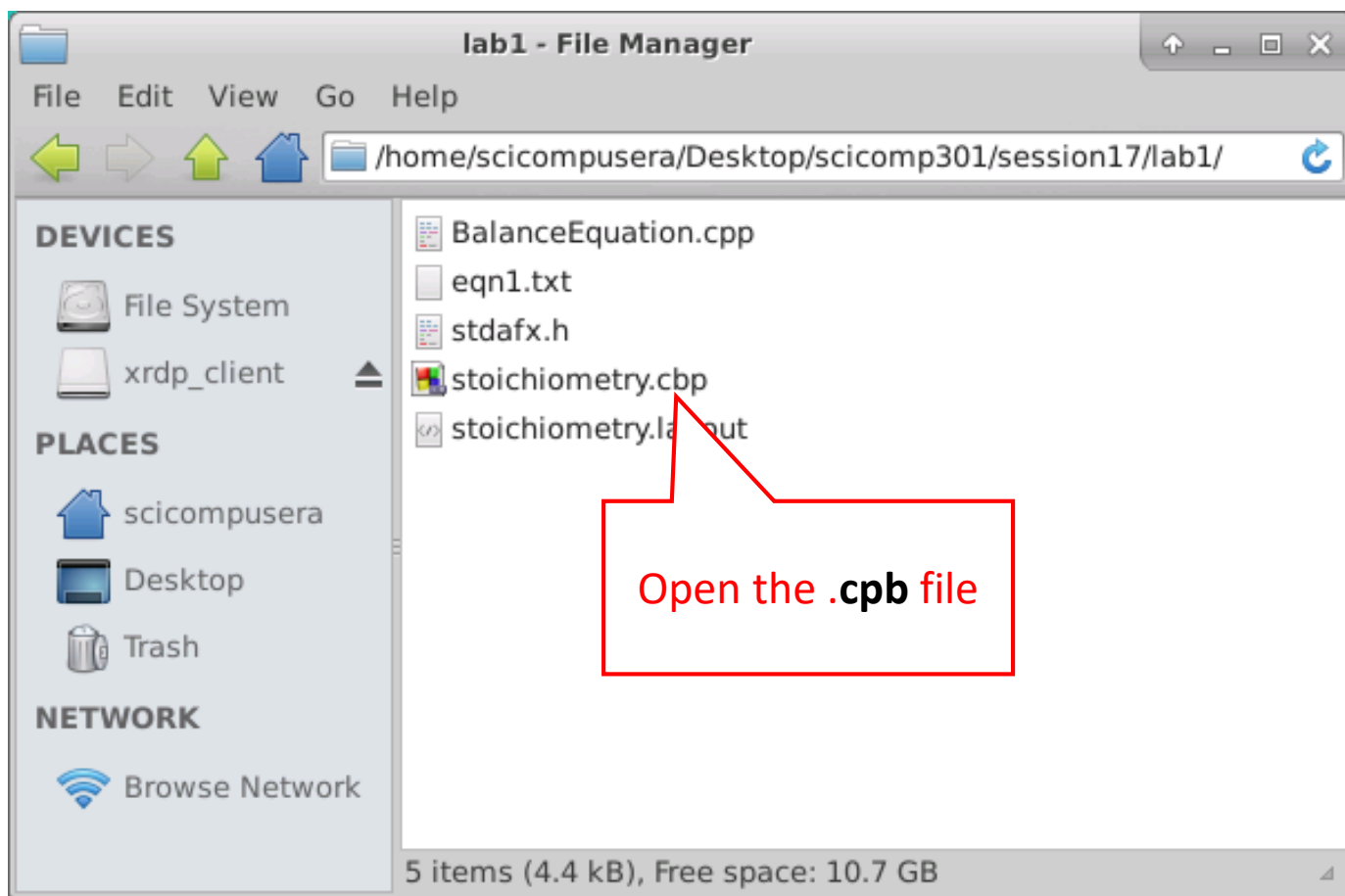


For the Oxygen atom these coefficients are taken from 3rd row in matrix

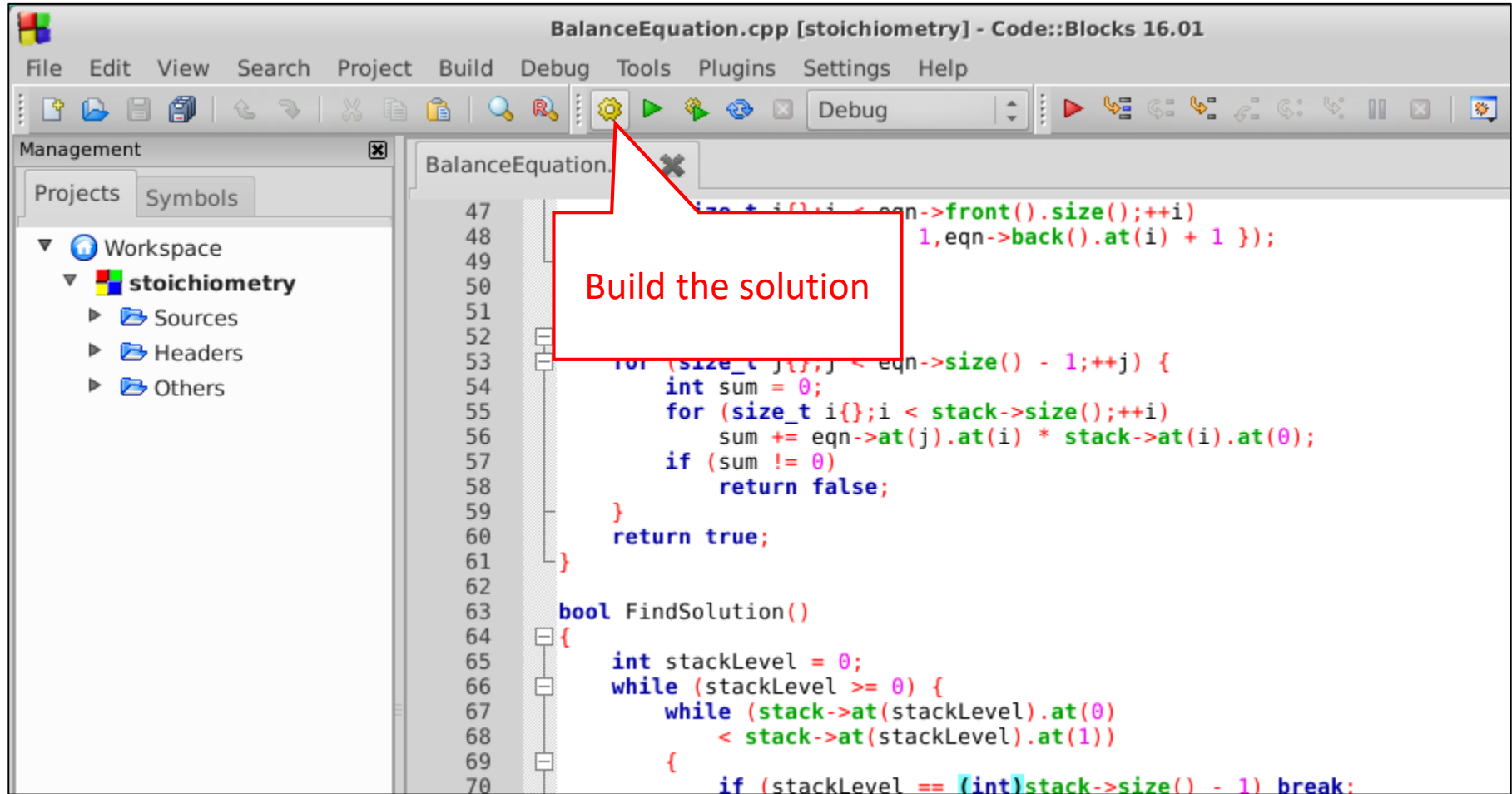
$$(4)x_0 + (2)x_1 + (0)x_2 + (0)x_3 + (-2)x_4 + (-1)x_5 = 0$$

- The current search values for each x_n are multiplied by each **column** coefficient in each **row** in the equation matrix – if the sum for **every row == 0**, then we have a *potential* solution!
- The final solution is where $\sum x_n$ has the lowest value (**POAC**)

Open Lab 1 – Stoichiometry



Build Lab 1 – Stoichiometry



Project Folder & File Structure

scicompuser#/~/

Desktop/scicomp301/

session17/lab1/

eqn1.txt

eqn2.txt

bin/

Debug/

Release/

stoichiometry

stoichiometry

In Linux only **folders** end with a forward slash /

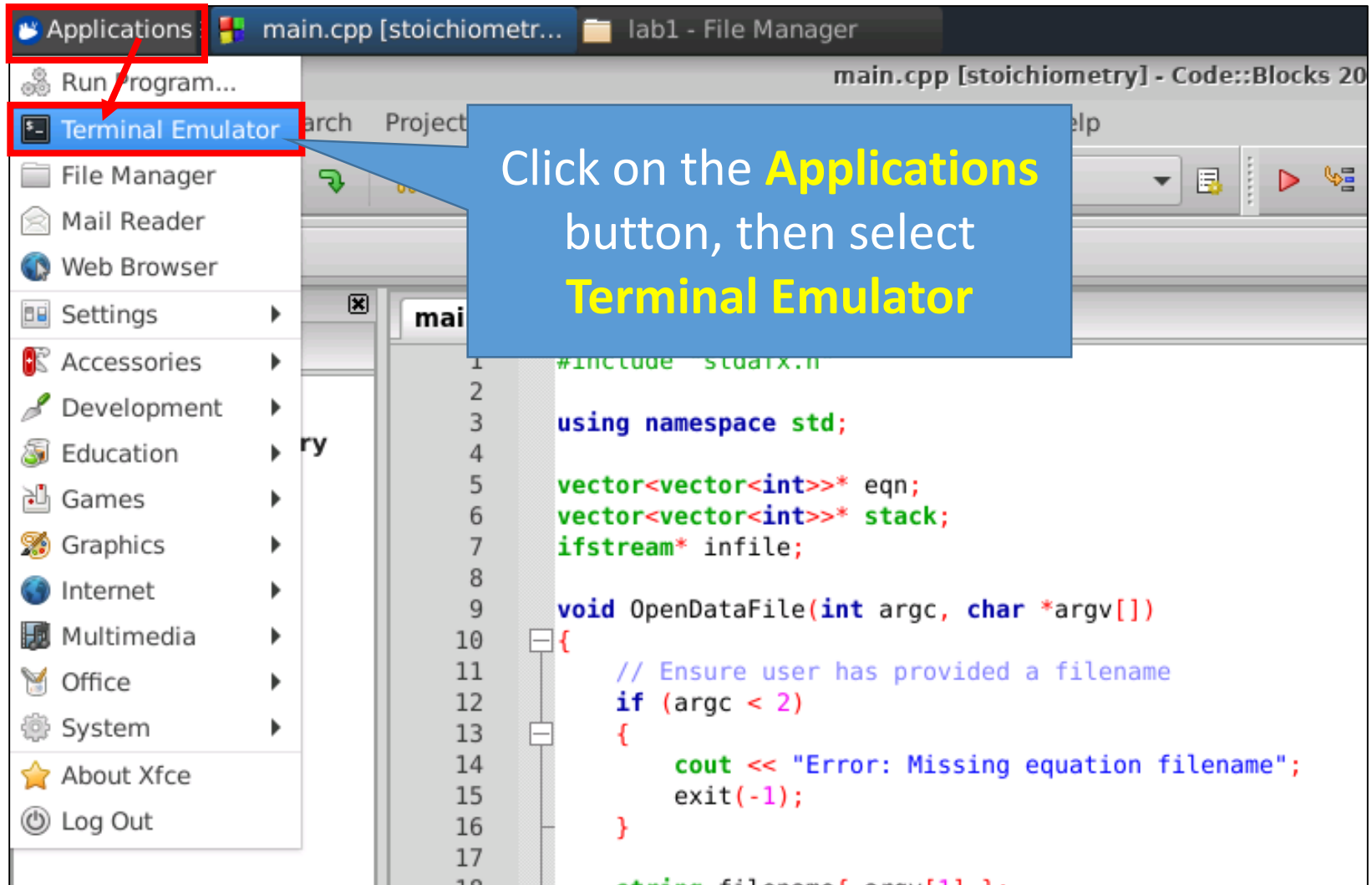
A **parent** folder can be specified using the **..** wildcard

The program **stoichiometry** is in the Debug/ folder

The **current** folder can be specified using the **.** wildcard

<i>app name</i>	<i>data file path</i>
./stoichiometry	../../eqn1.txt

Opening A Linux Terminal



Run Lab 1 – Stoichiometry

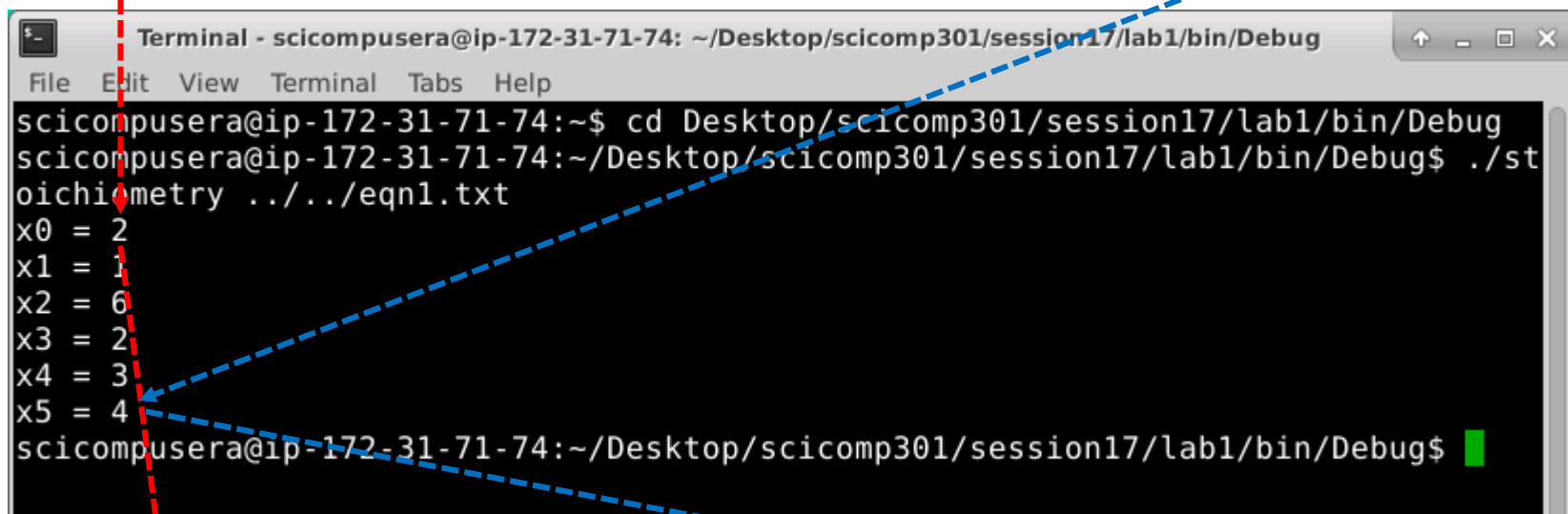
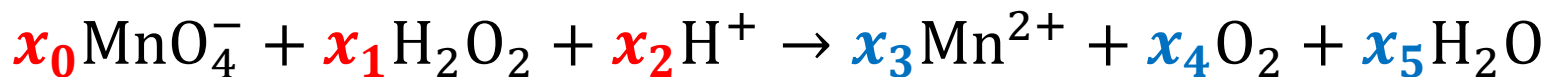
```
Terminal - scicompusera@ip-172-31-71-74: ~/Desktop/scicomp301/session17/lab1/bin/Debug
File Edit View Terminal Tabs Help
scicompusera@ip-172-31-71-74:~$ cd Desktop/scicomp301/session17/lab1/bin/Debug
scicompusera@ip-172-31-71-74:~/Desktop/scicomp301/session17/lab1/bin/Debug$ ./stoichiometry ../../eqn1.txt
x0 = 2
x1 = 1
x2 = 6
x3 = 2
x4 = 3
x5 = 4
scicompusera@ip-172-31-71-74:~/Desktop/scicomp301/session17/lab1/bin/Debug$
```

cd Desktop/scicomp301/session17/lab1/bin/Debug
./stoichiometry ../../eqn1.txt

Press **ENTER** after each line and **TAB** to auto-complete names

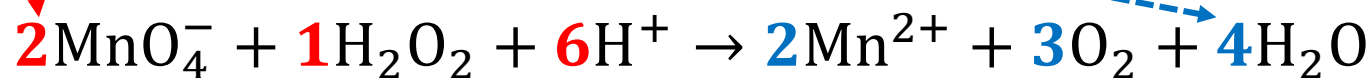
Remember Linux is case sensitive!

Verify Chemical Equation 1

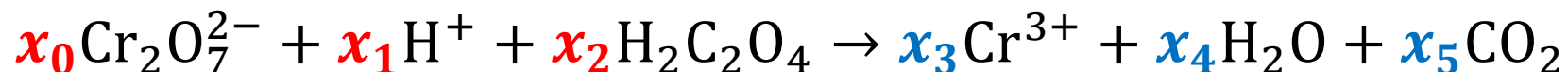


Terminal - scicompusera@ip-172-31-71-74: ~/Desktop/scicomp301/session17/lab1/bin/Debug

```
File Edit View Terminal Tabs Help
scicompusera@ip-172-31-71-74:~$ cd Desktop/scicomp301/session17/lab1/bin/Debug
scicompusera@ip-172-31-71-74:~/Desktop/scicomp301/session17/lab1/bin/Debug$ ./stoichiometry ../../eqn1.txt
x0 = 2
x1 = 1
x2 = 6
x3 = 2
x4 = 3
x5 = 4
scicompusera@ip-172-31-71-74:~/Desktop/scicomp301/session17/lab1/bin/Debug$
```



Encoding Chemical Equation 2



A Row for Every Element

Matrix Size (Rows=6 Cols=6)

Six Rows:

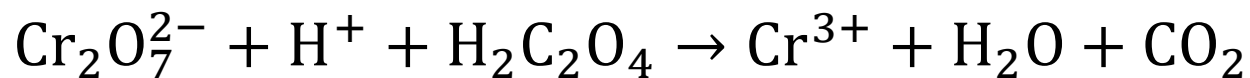
1. Chromium (Cr)
2. Oxygen (O)
3. Hydrogen (H)
4. Carbon (C)
5. Charges
6. Max Search Value per Term

A Column for Every Molecule

Six Columns:

1. x_0
2. x_1
3. x_2
4. x_3
5. x_4
6. x_5

Encoding Chemical Equation 2



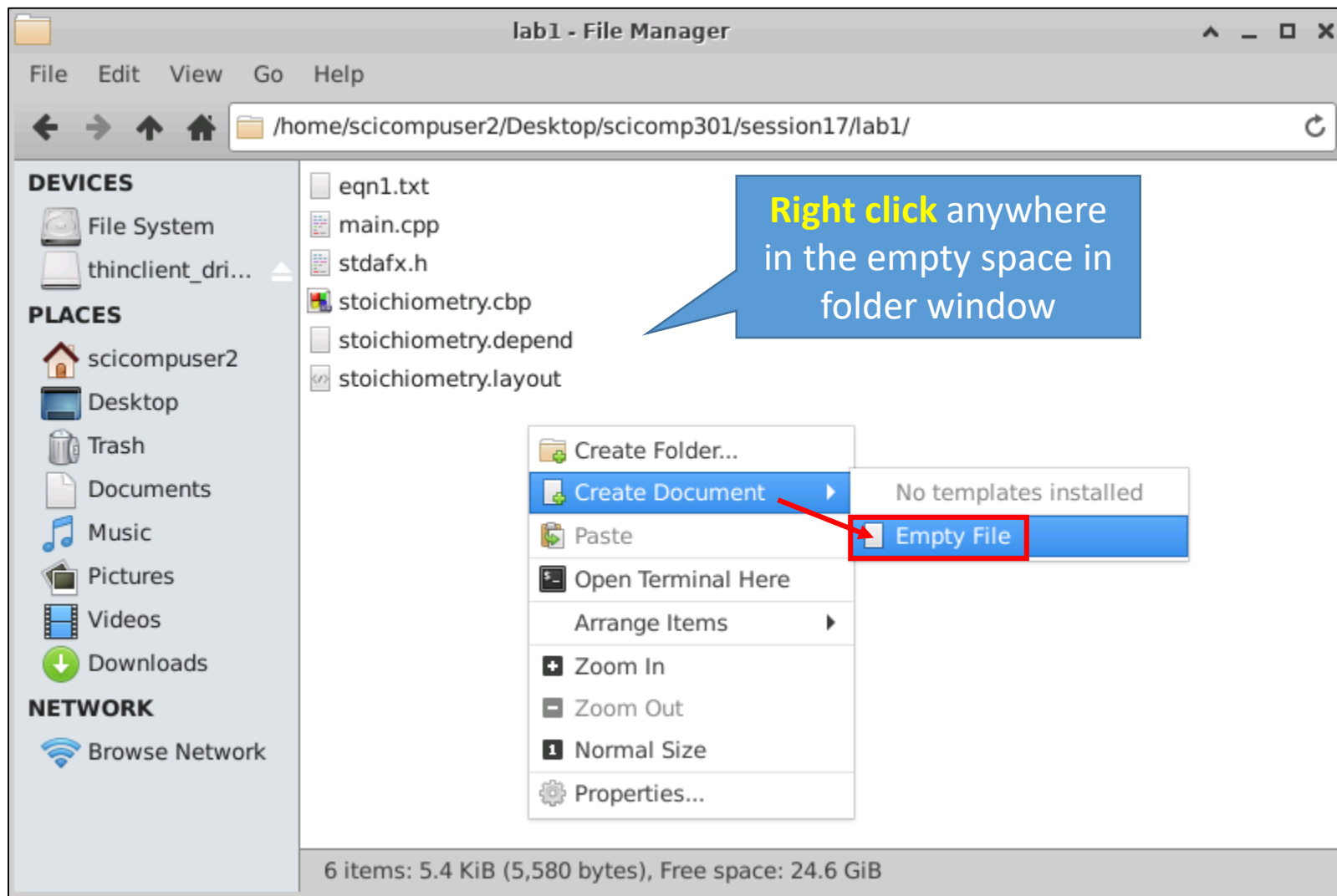
Reactant values are copied verbatim

Chromium (Cr) →
 Oxygen (O) →
 Hydrogen (H) →
 Carbon (C) →
 Charges (+/-) →
 Max Value →

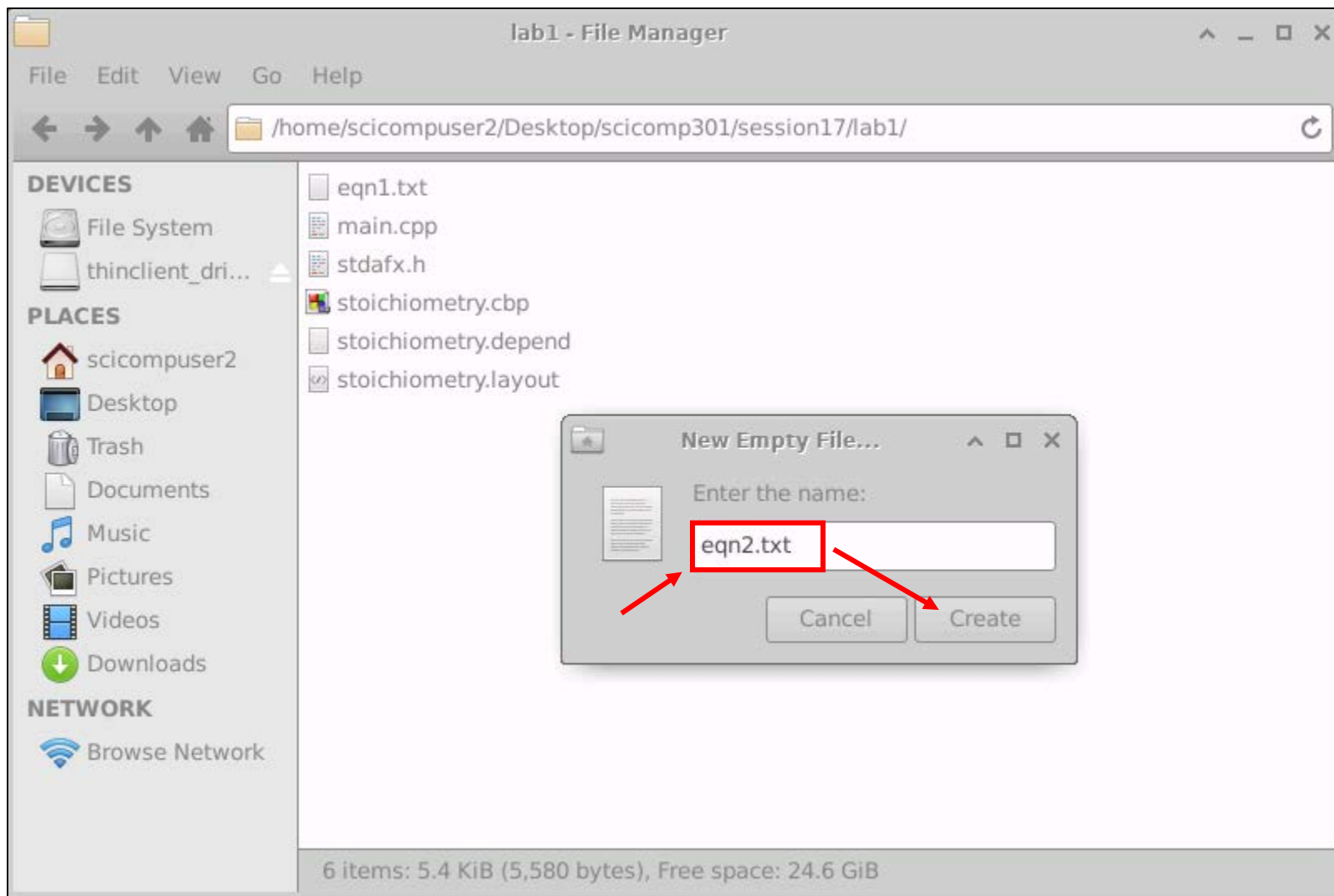
eqn2.txt ~/scicomp-labs...						
File	Edit	View	Search	Tools	Documents	
6	6					
2	0	0	-1	0	0	
7	0	4	0	-1	-2	
0	1	2	0	-2	0	
0	0	2	0	0	-1	
-2	1	0	-3	0	0	
9	9	9	9	9	9	

Product values are **negated !!**

Encoding Chemical Equation 2

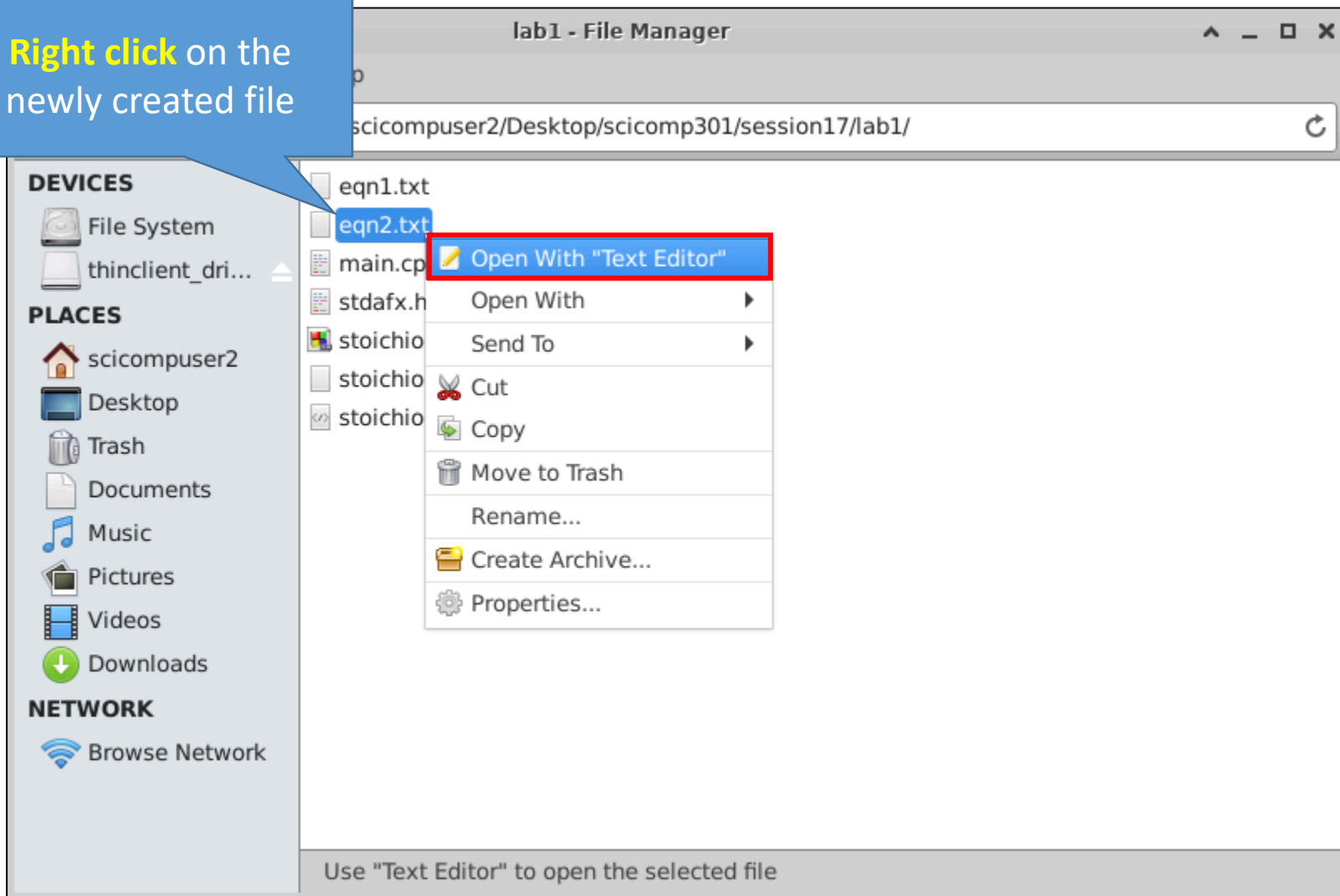


Encoding Chemical Equation 2

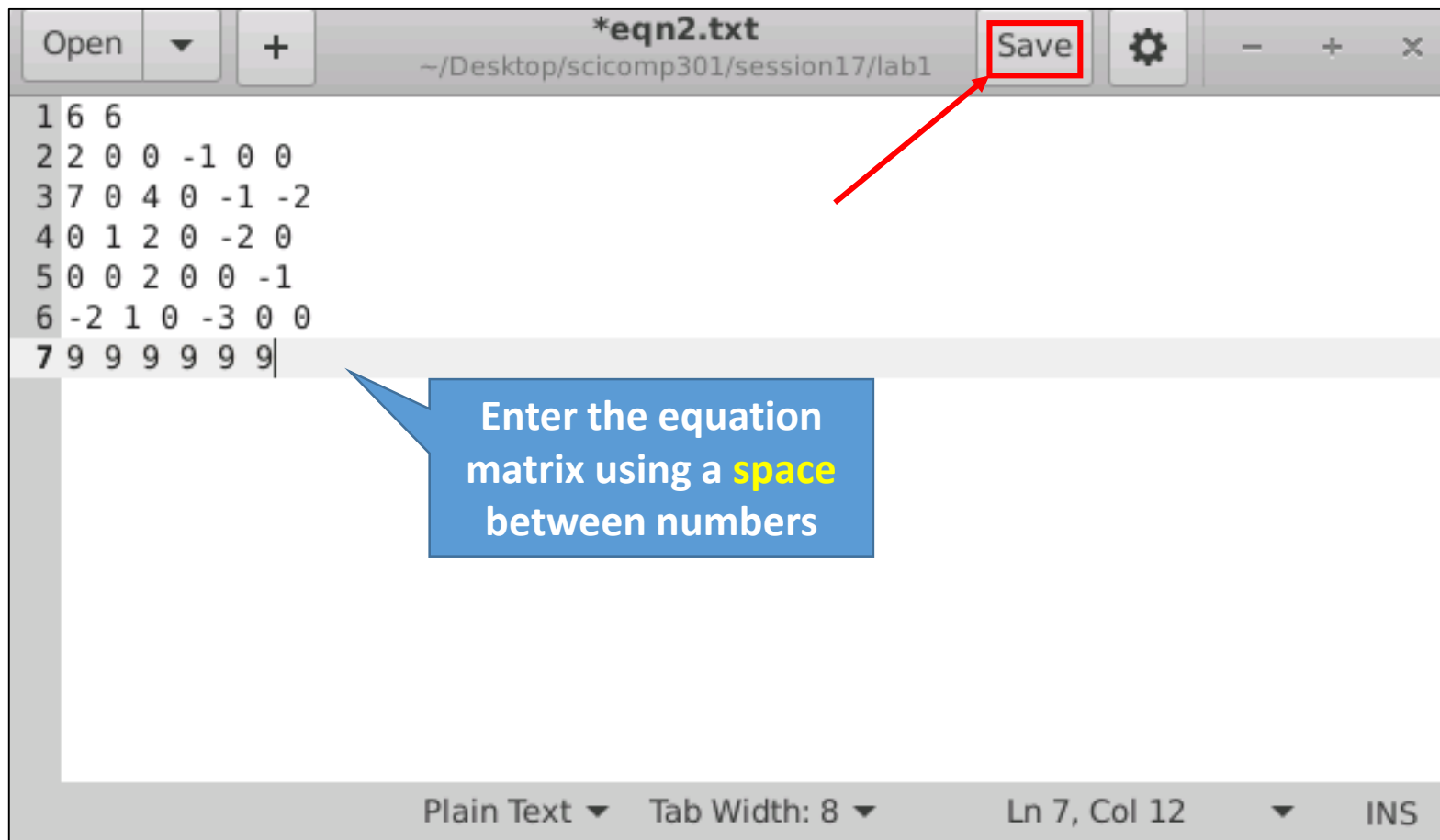


Encoding Chemical Equation 2

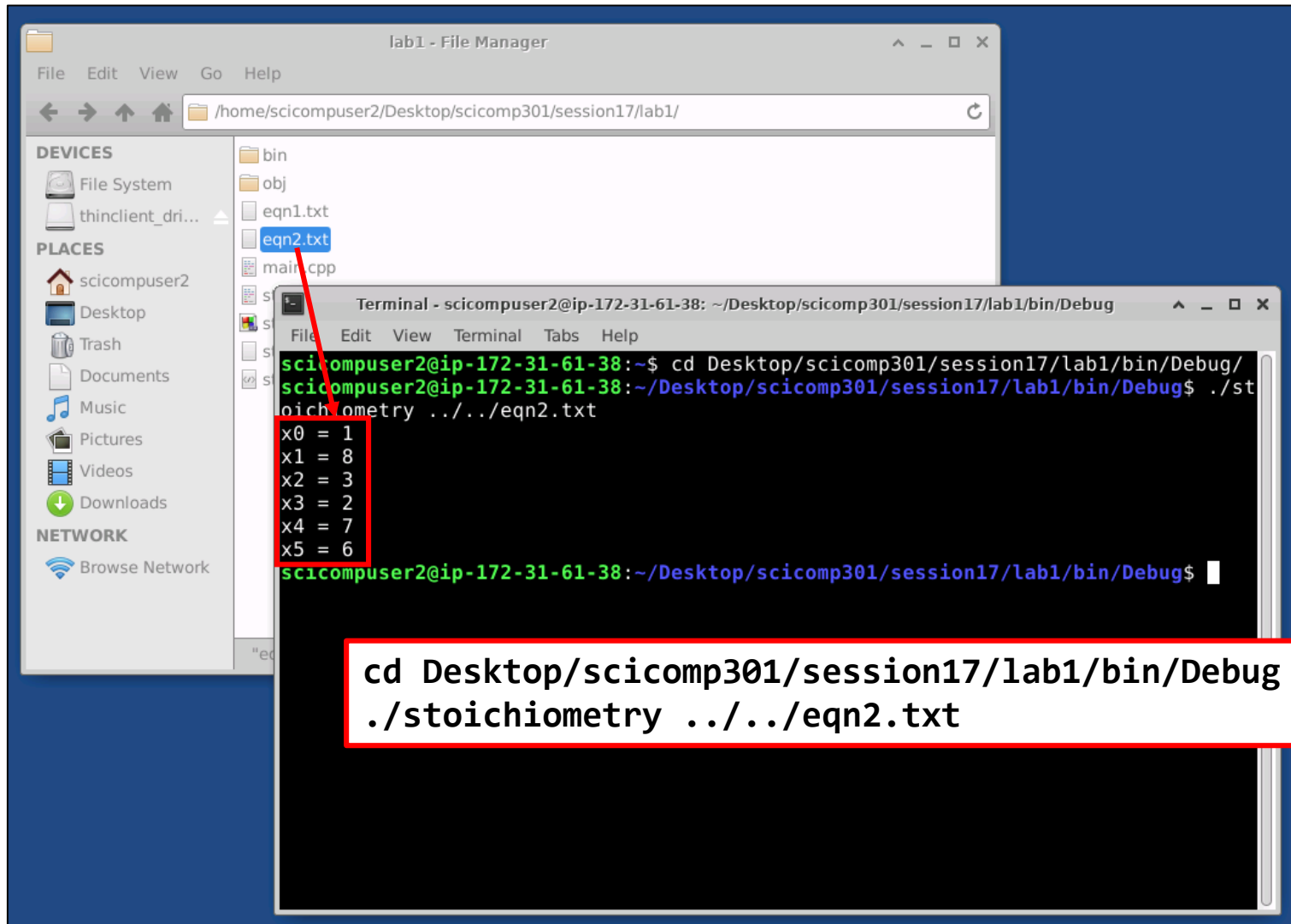
Right click on the newly created file



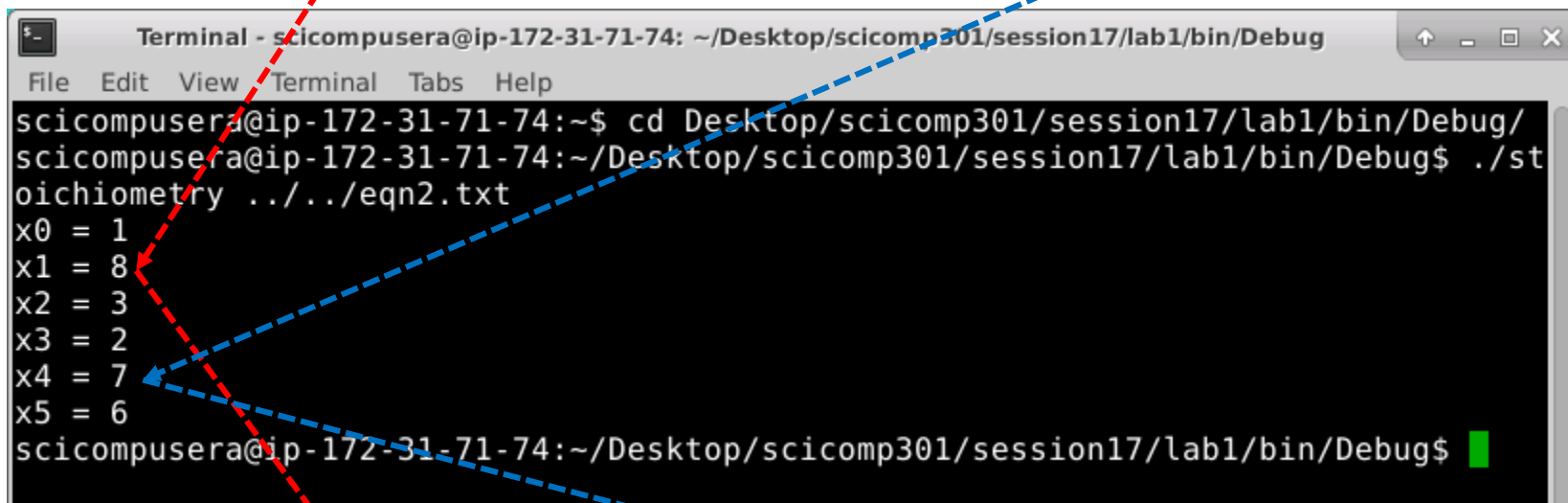
Encoding Chemical Equation 2



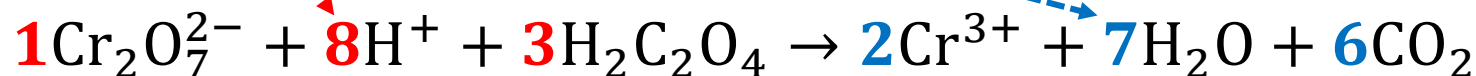
Balancing Chemical Equation 2



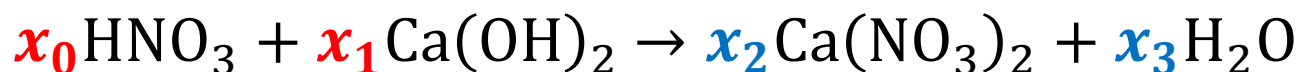
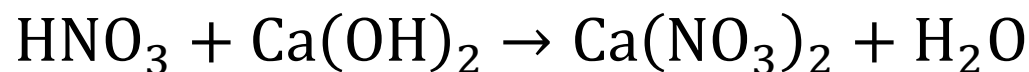
Verify Chemical Equation 2



```
Terminal - scicompusera@ip-172-31-71-74: ~/Desktop/scicomp301/session17/lab1/bin/Debug
File Edit View Terminal Tabs Help
scicompusera@ip-172-31-71-74:~$ cd Desktop/scicomp301/session17/lab1/bin/Debug/
scicompusera@ip-172-31-71-74:~/Desktop/scicomp301/session17/lab1/bin/Debug$ ./st
oichiometry ../../eqn2.txt
x0 = 1
x1 = 8
x2 = 3
x3 = 2
x4 = 7
x5 = 6
scicompusera@ip-172-31-71-74:~/Desktop/scicomp301/session17/lab1/bin/Debug$
```



Encoding Chemical Equation 3



There is no row for **charges** in this equation

Hydrogen (H) →
Nitrogen (N) →
Oxygen (O) →
Calcium (Ca) →
Max Value →

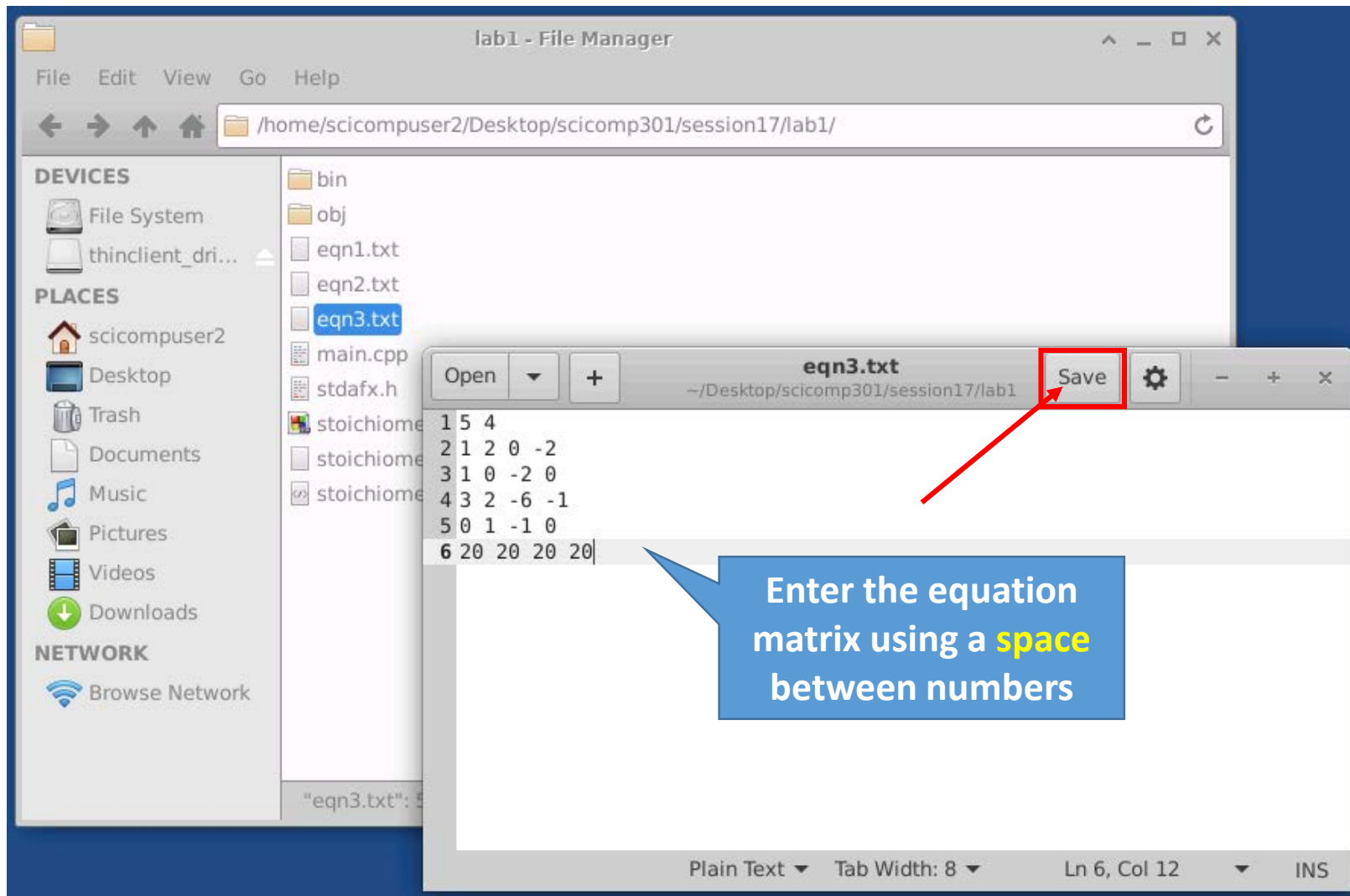
The image shows a screenshot of a code editor window. The title bar at the top displays 'eqn3.txt' and the file path '~/scicomp-labs...'. The menu bar includes 'File', 'Edit', 'View', 'Search', 'Tools', and 'Document'. The main text area contains a 5x4 matrix of numbers:

5	4		
1	2	0	-2
1	0	-2	0
3	2	-6	-1
0	1	-1	0
20	20	20	20

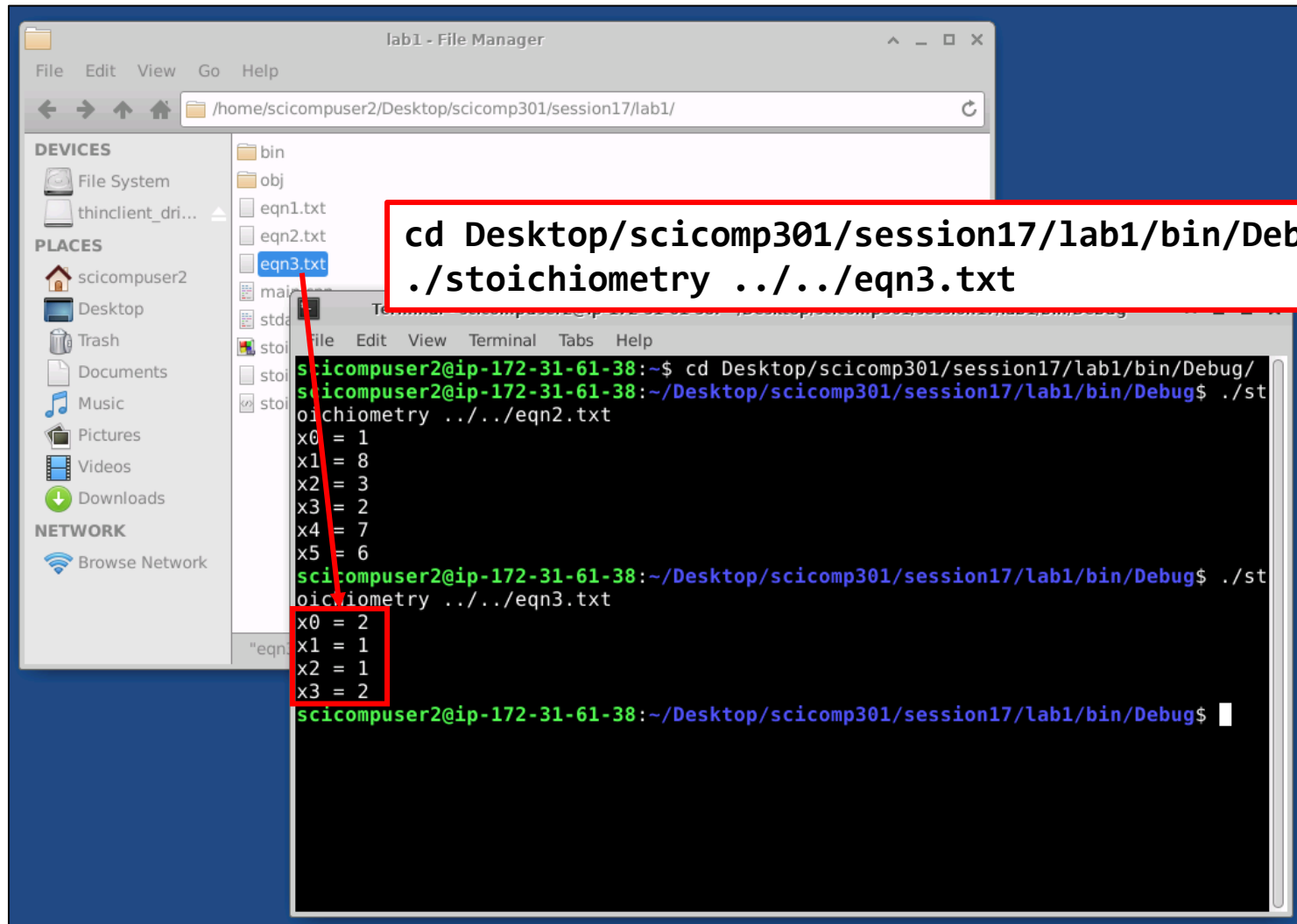
A red dashed arrow points from the 'Open' button in the title bar to the first element '5' of the matrix. A blue dashed arrow points from the '+' button in the title bar to the element '-6' in the matrix. A black-bordered box on the right side of the image contains the text: 'Watch the multiplier for parentheses!'.

Watch the multiplier for **parentheses!**

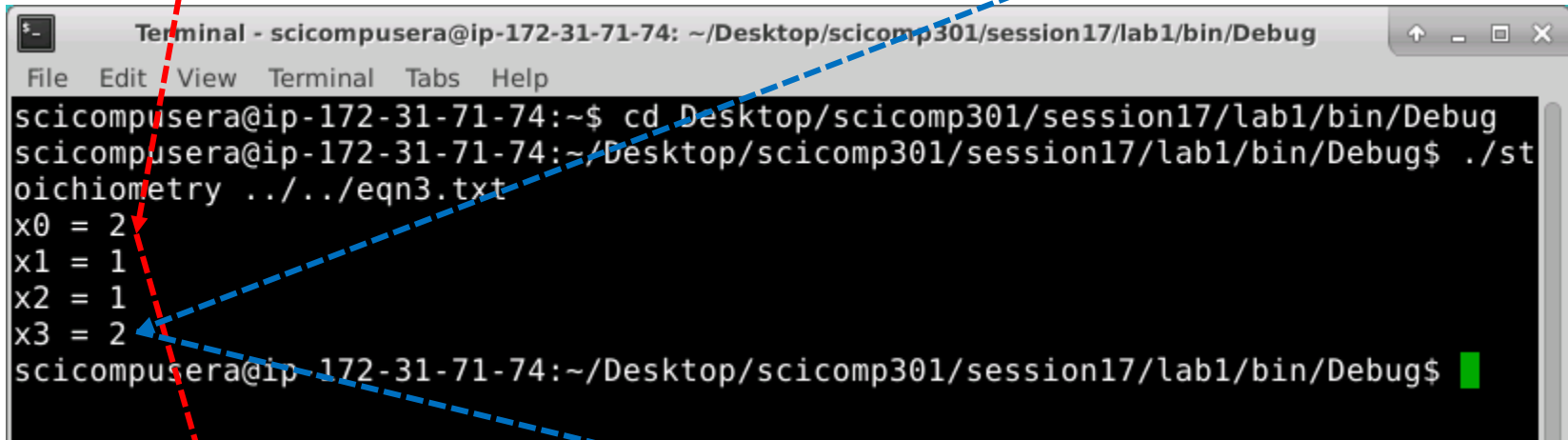
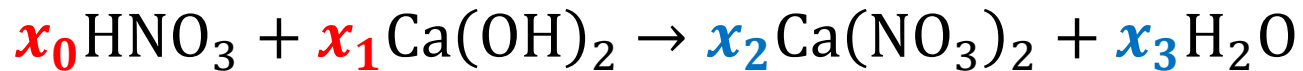
Encoding Chemical Equation 3



Balancing Chemical Equation 3



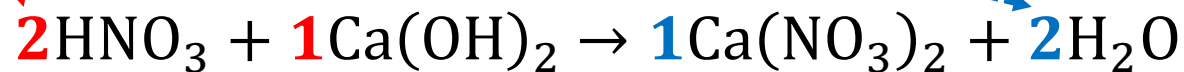
Verify Chemical Equation 3



A terminal window titled "Terminal - scicompusera@ip-172-31-71-74: ~/Desktop/scicomp301/session17/lab1/bin/Debug". The window contains the following text:

```
scicompusera@ip-172-31-71-74:~$ cd Desktop/scicomp301/session17/lab1/bin/Debug
scicompusera@ip-172-31-71-74:~/Desktop/scicomp301/session17/lab1/bin/Debug$ ./stoichiometry ../../eqn3.txt
x0 = 2
x1 = 1
x2 = 1
x3 = 2
scicompusera@ip-172-31-71-74:~/Desktop/scicomp301/session17/lab1/bin/Debug$
```

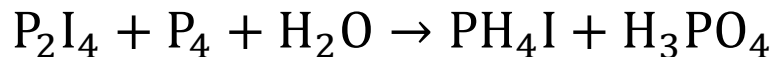
Two dashed arrows originate from the terminal output. A red dashed arrow points from the value "2" next to x_0 to the coefficient "2" in the balanced equation below. A blue dashed arrow points from the value "1" next to x_1 to the coefficient "1" in the balanced equation below. Another blue dashed arrow points from the value "1" next to x_2 to the coefficient "1" in the balanced equation below. A final blue dashed arrow points from the value "2" next to x_3 to the coefficient "2" in the balanced equation below.



Avoid Writing Brittle Code

```
static void NestedForLoops()
{
    for (int x0 = 0; x0 < 50; x0++)
    {
        for (int x1 = 0; x1 < 50; x1++)
        {
            for (int x2 = 0; x2 < 50; x2++)
            {
                for (int x3 = 0; x3 < 50; x3++)
                {
                    for (int x4 = 0; x4 < 50; x4++)
                    {
                        for (int x5 = 0; x5 < 50; x5++)
                        {
                            for (int x6 = 0; x6 < 50; x6++)
                            {
                                // Evaluate objective function
                            }
                        }
                    }
                }
            }
        }
    }
}
```

- We could use nested **for()** loops, but what if number of unknowns (x_n) changes?
- Objective function would be evaluated **50⁷** times =
781,250,000,000
- No guarantee that molar coefficients $1 \leq x_n \leq 50$



Use Data Structures vs. Brittle Code

```
vector<vector<int>>*> stack;
```

<i>Unknown</i>	Index	Current Value	Max Value
x_0	0	1	9
x_1	1	1	9
x_2	2	1	9
x_3	3	1	9
x_4	4	1	9
x_5	5	1	9

← stack level = 4

Our **stack** is a **vector of vectors**, with a **row** for each **unknown** and each row having **two** columns.

We also keep track of the current **stack level**

Use Data Structures vs. Brittle Code

Level	Range								
x_0	1	2	3	4	5	6	7	8	9
x_1	1	2	3	4	5	6	7	8	9
x_2	1	2	3	4	5	6	7	8	9
x_3	1	2	3	4	5	6	7	8	9
x_4	1	2	3	4	5	6	7	8	9
x_5	1	2	3	4	5	6	7	8	9

- At each iteration, the current level moves to the next position in its range, and checks the constraints

Level	Range								
x_0	1	2	3	4	5	6	7	8	9
x_1	1	2	3	4	5	6	7	8	9
x_2	1	2	3	4	5	6	7	8	9
x_3	1	2	3	4	5	6	7	8	9
x_4	1	2	3	4	5	6	7	8	9
x_5	1	2	3	4	5	6	7	8	9

- If the equation balances, but a new **minimum** atom count (sum) is found, save the current value for each unknown, and that sum becomes the new minimum

Use Data Structures vs. Brittle Code

Level	Range								
x_0	1	2	3	4	5	6	7	8	9
x_1	1	2	3	4	5	6	7	8	9
x_2	1	2	3	4	5	6	7	8	9
x_3	1	2	3	4	5	6	7	8	9
x_4	1	2	3	4	5	6	7	8	9
x_5	1	2	3	4	5	6	7	8	9

Level	Range								
x_0	1	2	3	4	5	6	7	8	9
x_1	1	2	3	4	5	6	7	8	9
x_2	1	2	3	4	5	6	7	8	9
x_3	1	2	3	4	5	6	7	8	9
x_4	1	2	3	4	5	6	7	8	9
x_5	1	2	3	4	5	6	7	8	9

Level	Range								
x_0	1	2	3	4	5	6	7	8	9
x_1	1	2	3	4	5	6	7	8	9
x_2	1	2	3	4	5	6	7	8	9
x_3	1	2	3	4	5	6	7	8	9
x_4	1	2	3	4	5	6	7	8	9
x_5	1	2	3	4	5	6	7	8	9

- Just as a car odometer rolls over when a digit range reaches its limit, we increment the prior level's position and reset the current level to 1

- The program ends when every level in the stack has reached the **limit** of its *individually* assigned range

Use Data Structures vs. Brittle Code

```
bool FindSolution()
{
    int stackLevel = 0;
    while (stackLevel >= 0) {
        while (stack->at(stackLevel).at(0)
               < stack->at(stackLevel).at(1))
        {
            if (stackLevel == (int)stack->size() - 1) break;
            stackLevel++;
            stack->at(stackLevel).at(0) = 1;

            if (IsSolution())
                return true;

            stack->at(stackLevel).at(0)++;
            while (stack->at(stackLevel).at(0)
                   == stack->at(stackLevel).at(1))
            {
                stackLevel--;
                if (stackLevel < 0) break;
                stack->at(stackLevel).at(0)++;
            }
        }
        return false;
    }
}
```

Invoke the
objective
function

Use the next
value in this
level's range

Find the lowest level
in the stack which has
not yet enumerated
its full range

Walk up the stack,
skipping all parent
levels that have
already completed
their range

Using Data Structures vs. Brittle Code

```
void LoadEquation()
{
    int constraints, unknowns;
    *infile >> constraints >> unknowns;

    eqn = new vector<vector<int>>(constraints);
    vector<int> constraint(unknowns);

    for (auto& e : *eqn) {
        for (auto& c : constraint)
            *infile >> c;
        e = constraint;
    }
}
```

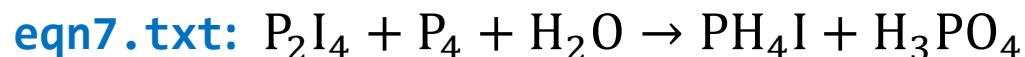
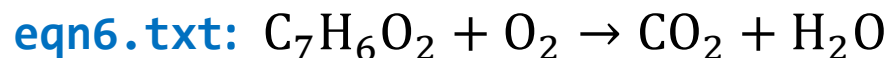
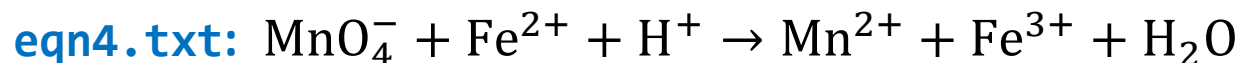
To read from a
file use the
stream insertion
>> operator

A solution is valid only if the
convolution of every matrix
row with every stack row == 0

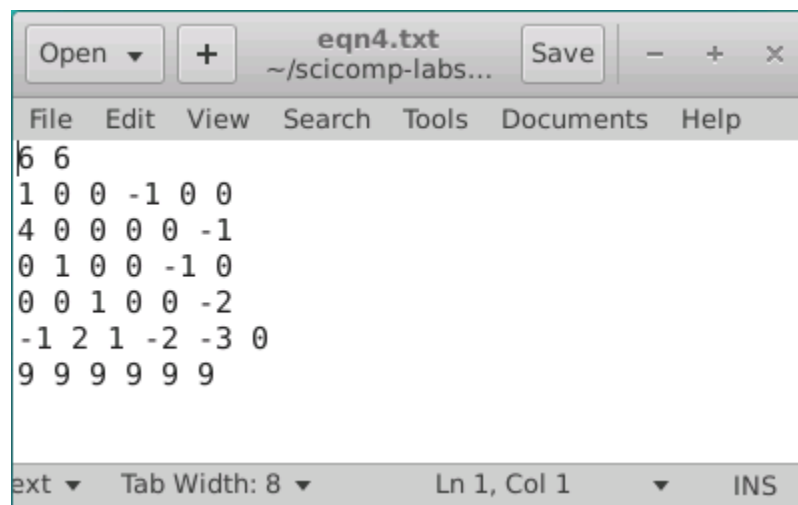
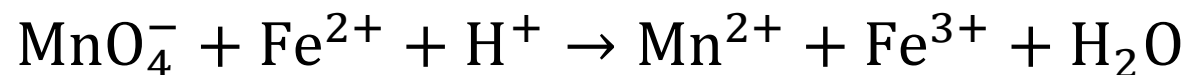
```
bool IsSolution()
{
    for (size_t j{}; j < eqn->size() - 1; ++j) {
        int sum = 0;
        for (size_t i{}; i < stack->size(); ++i)
            sum += eqn->at(j).at(i) * stack->at(i).at(0);
        if (sum != 0)
            return false;
    }
    return true;
}
```

Balancing Ionic Equations

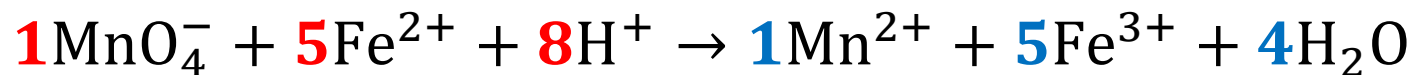
- Pick **one** of the following equations to balance using the **stoichiometry** application
- In the Lab 1 folder, **create** a text file that encodes your chosen equation using the corresponding filename



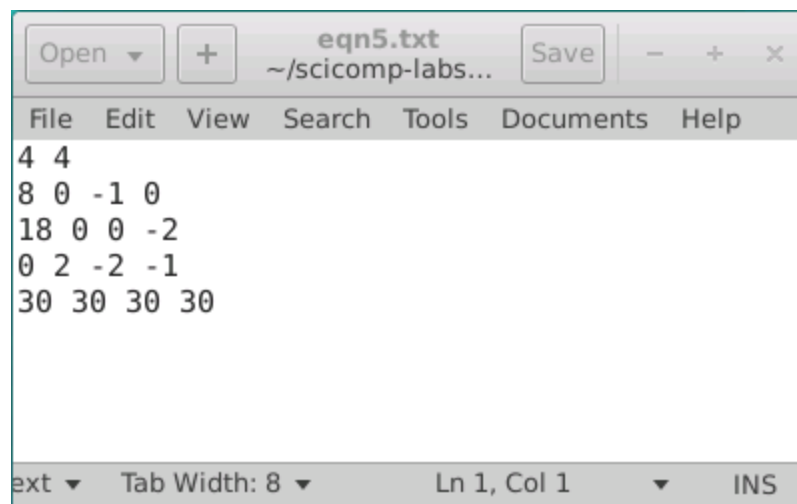
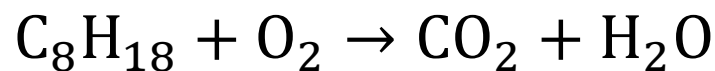
Balancing Chemical Equation 4



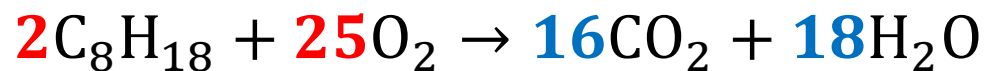
```
eqn4.txt
~/scicomp-labs...
File Edit View Search Tools Documents Help
6 6
1 0 0 -1 0 0
4 0 0 0 0 -1
0 1 0 0 -1 0
0 0 1 0 0 -2
-1 2 1 -2 -3 0
9 9 9 9 9 9
ext Tab Width: 8 Ln 1, Col 1 INS
```



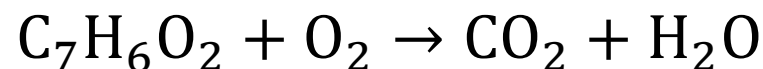
Balancing Chemical Equation 5



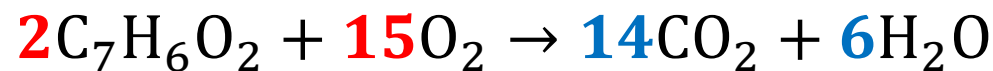
```
eqn5.txt
~/scicomp-labs...
Open + Save - + x
File Edit View Search Tools Documents Help
4 4
8 0 -1 0
18 0 0 -2
0 2 -2 -1
30 30 30 30
ext Tab Width: 8 Ln 1, Col 1 INS
```



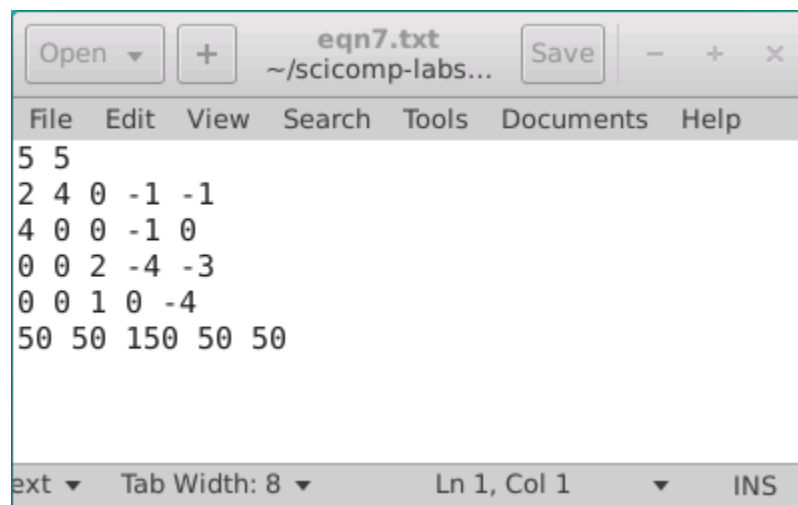
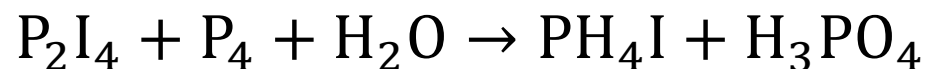
Balancing Chemical Equation 6



```
4 4
7 0 -1 0
6 0 0 -2
2 2 -2 -1
20 20 20 20
```



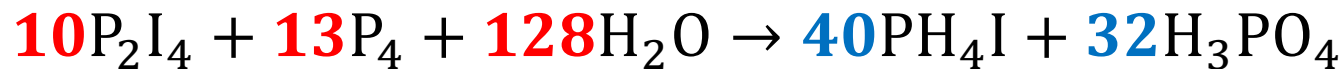
Balancing Chemical Equation 7



The screenshot shows a text editor window titled "eqn7.txt" with the following content:

```
5 5
2 4 0 -1 -1
4 0 0 -1 0
0 0 2 -4 -3
0 0 1 0 -4
50 50 150 50 50
```

The window has a menu bar with File, Edit, View, Search, Tools, Documents, and Help. The status bar at the bottom shows "ext", "Tab Width: 8", "Ln 1, Col 1", and "INS".



Diphosphorus tetraiodide

Now you know...

<https://www.webqc.org/balance.php>

- We can encode a chemical equation into a **matrix**, where the **rows** represent the element constraints for reactants and products and the **columns** represent the unknown coefficients of the molecules
- We can transform **balancing ionic equations** to a linear program, which can be solved using brute-force if the search space is sufficiently small
- Even relatively simple equations can exponentially increase the search space & using nested **for()** loops makes the code too **brittle**
- It is better to use the **Simplex** method, and normalize any rational solutions back to integers for the final answer

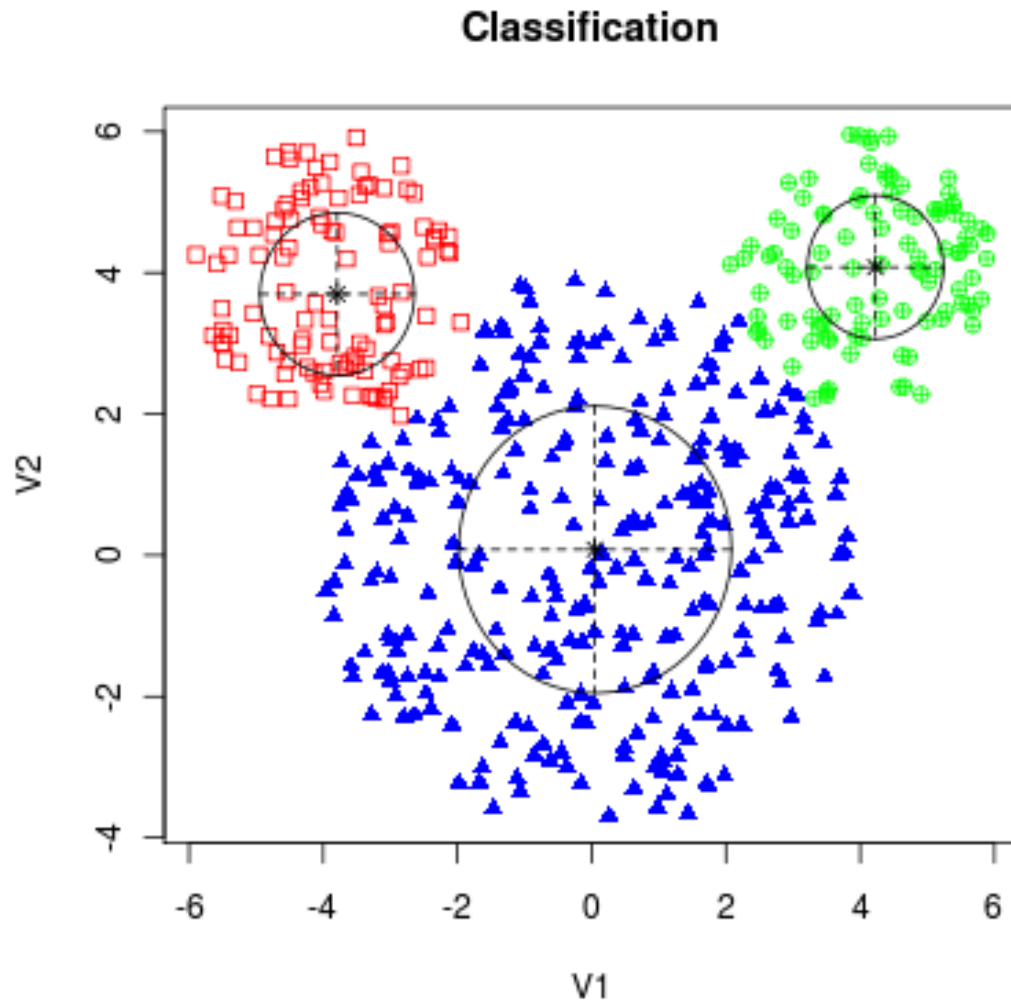
Session Goals

- Study an efficient algorithm for grouping similar **data sets**
- Gain an appreciation for **k-means Clustering** – an algorithm that iterates by **assigning data points to a cluster**
 - The goal is to programmatically assign data points that are “near” each other to the same **cluster** so we can perform group statistics on data that “belongs” together
 - Learn the limitations of k-means and visualize the dangers of “over fitting” data
 - Envision how we can use a **mean (μ) distance cut-off** to programmatically identify **data outliers**

k-means Clustering

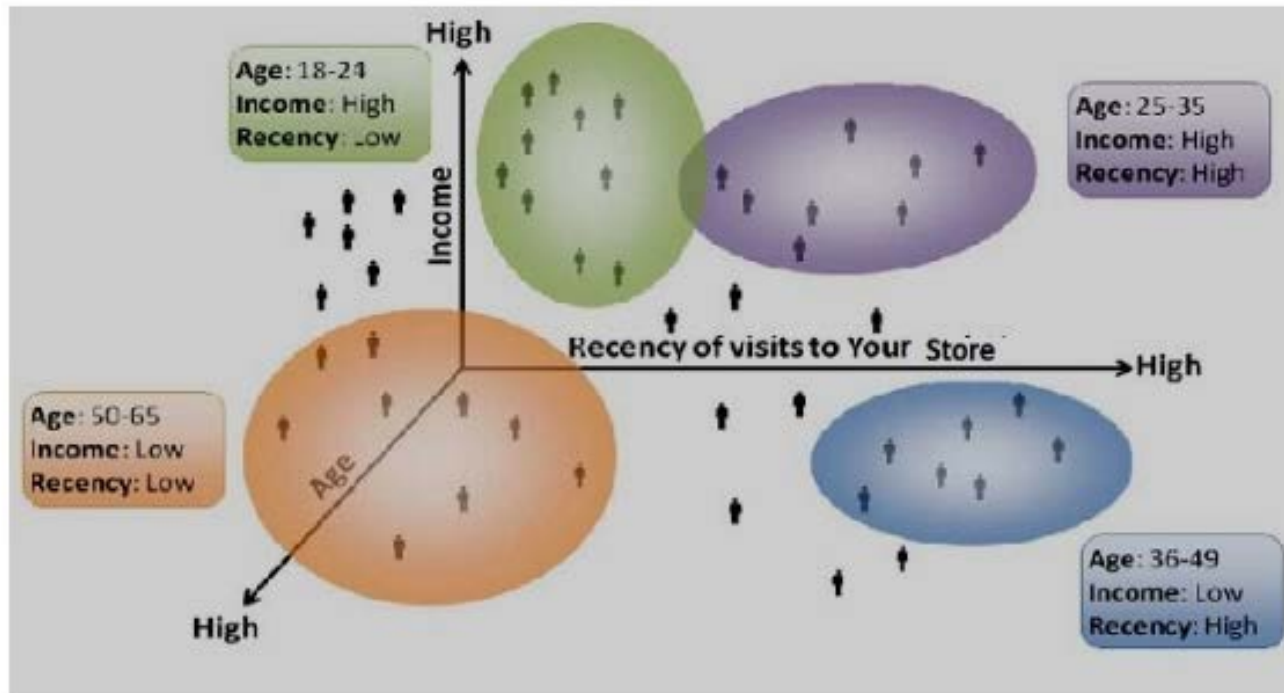
- k -means clustering aims to partition n observations into k clusters (where each cluster has its own μ point)
- Each observation is assigned to the cluster whose μ point **is closest** to that observation's point
- The problem is computationally difficult (NP-hard); however, there are efficient heuristic algorithms that are commonly employed and **converge quickly to a local optimum**
- The term " k -means" was first used by James MacQueen in 1967, though the idea goes back to Hugo Steinhaus in 1957
- A more efficient version was published Hartigan and Wong in **1979**

k-means Clustering



k-means Clustering

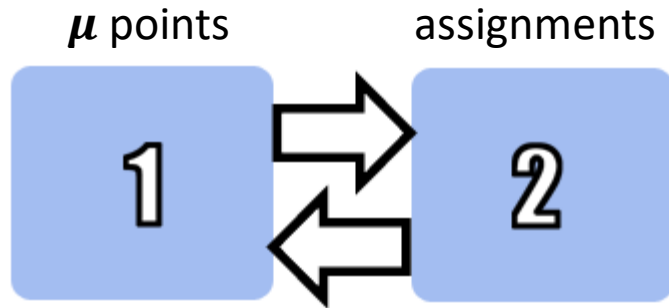
Example - Clusters using Age, Income & Recency



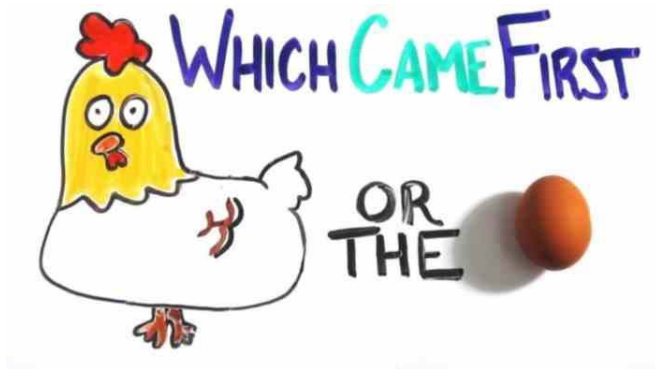
k-means Clustering

- A **one-to-many** mapping between **clusters** and **data points**
 - A **cluster** is a set of one or more unique data points
 - Every **data point** must be assigned to **only one** cluster at a time
 - Data points are initially assigned to clusters in a round-robin fashion
- The μ point of each **cluster** is its geometric center (**centroid**) calculated by taking the average value (in each dimension) of **only** the **data points** assigned to that cluster
- Assignments *can* change the μ points... which then *can* change the assignments... which then *can* change the μ points... which then *can* change the assignments... which then *can*...

k-means Clustering



$$G_{\mu\nu} + \Lambda T_{\mu\nu} = \frac{8\pi G}{c^4} T_{\mu\nu}$$



Spacetime tells matter how to move;
matter tells spacetime how to curve.

— John Archibald Wheeler —

k-means Clustering

- The algorithm uses **two** phases per round:
 1. For each **cluster**, a new centroid (μ point) is calculated using the current data point assignments
If the new $\mu \neq$ current μ then **move cluster** so current $\mu =$ new μ
 2. For each **data point**, find the cluster which has the closest μ point using Pythagorean distance formula
If necessary, **reassign the data point** to the new closest cluster - subject to constraint that every cluster must have one data point
- The algorithm repeats phases #1 and #2 until no cluster moves and no data point is reassigned
- When the algorithm reaches steady-state the cluster has “**converged**”

Open Lab 2 – kMeansClustering.h

```
class DataPoint
{
public:
    DataPoint() = default;
    DataPoint(double new_x, double new_y);
    double x{};
    double y{};
    Cluster* c{nullptr};
};
```

Every datapoint
belongs to just one
cluster

```
class Cluster
{
public:
    Cluster(int index);
    double x{};
    double y{};
    string clr{};
    int population{};
    double mean_distance{};
};
```

View Lab 2 – kMeansClustering.h

```
class DataPoint
{
public:
    DataPoint() = default;
    DataPoint(double new_x, double new_y);
    double x{};
    double y{};
    Cluster* c{nullptr};
};
```

Every **datapoint** has
an (x, y) Cartesian
coordinate

The μ point (center) of a
cluster is the average of
the coordinates (x, y) of
all the datapoints that
belong to that cluster

```
class Cluster
{
public:
    Cluster(int index, ...);
    double x{};
    double y{};
    string clr{};
    int population{};
    double mean_distance{};
};
```

Every **cluster** has a
center (x, y)
Cartesian coordinate

View Lab 2 – kMeansClustering.h

```
class DataPoint
{
public:
    DataPoint() = default;
    DataPoint(double new_x, double new_y);
    double x{};
    double y{};
    Cluster* c{nullptr};
};
```

Every cluster has a
distinct Allegro **color**

```
class Cluster
{
public:
    Cluster(int index);
    double x{};
    double y{};
    string clr{};
    int population{};
    double mean_distance{};
};
```

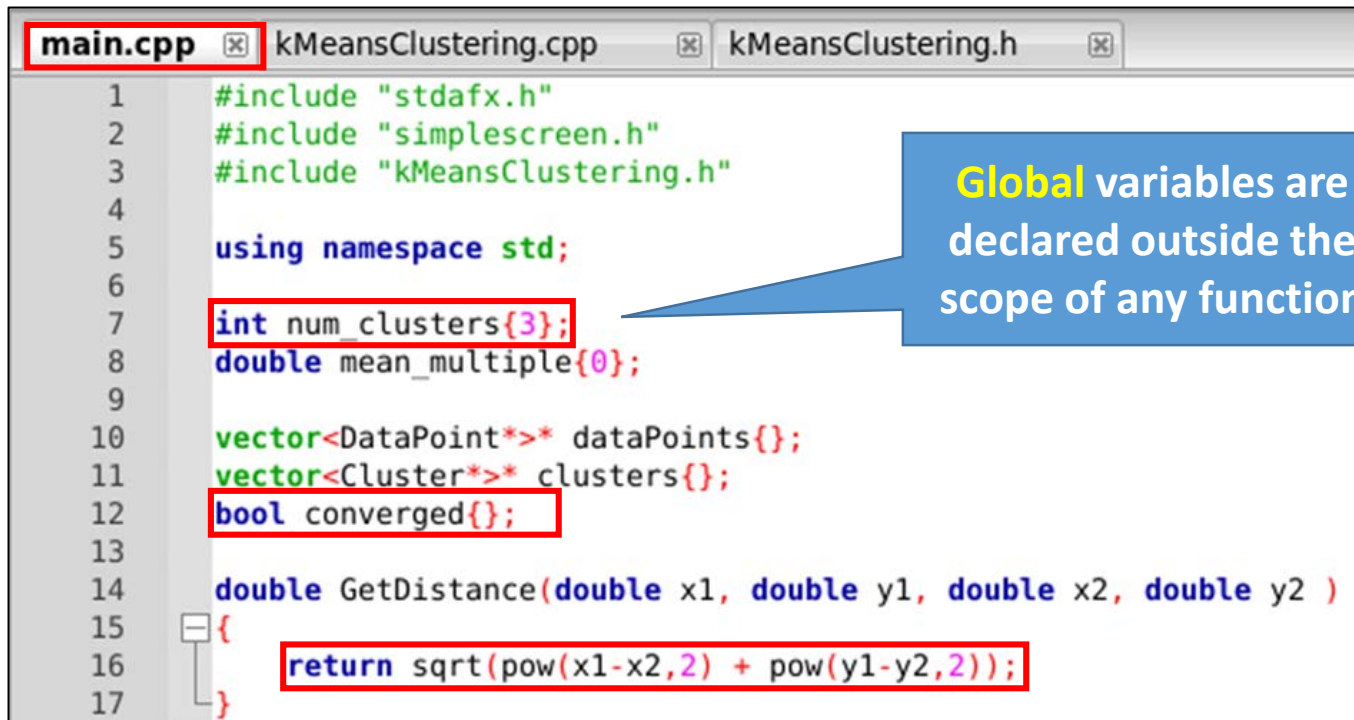
View Lab 2 – kMeansClustering.h

```
class DataPoint
{
public:
    DataPoint() = default;
    DataPoint(double new_x, double new_y);
    double x{};
    double y{};
    Cluster* c{nullptr};
};
```

The number of
datapoints belonging
to a cluster is its
population

```
class Cluster
{
public:
    Cluster(int index);
    double x{};
    double y{};
    string clr{};
    int population{};
    double mean_distance{};
};
```

Open Lab 2 – main.cpp



```
main.cpp x kMeansClustering.cpp x kMeansClustering.h x
1  #include "stdafx.h"
2  #include "simplescreen.h"
3  #include "kMeansClustering.h"
4
5  using namespace std;
6
7  int num_clusters{3};
8  double mean_multiple{0};
9
10 vector<DataPoint*> dataPoints{};
11 vector<Cluster*> clusters{};
12 bool converged{};
13
14 double GetDistance(double x1, double y1, double x2, double y2 )
15 {
16     return sqrt(pow(x1-x2,2) + pow(y1-y2,2));
17 }
```

Global variables are declared outside the scope of any function

Even k-means uses the Pythagorean theorem!

View Lab 2 – main.cpp

```
main.cpp kMeansClustering.cpp kMeansClustering.h
13
14 double GetDistance(double x1, double y1, double x2,
15 {
16     return sqrt(pow(x1-x2,2) + pow(y1-y2,2));
17 }
18
19 void InitDataPoints()
20 {
21     dataPoints = new vector<DataPoint*>();
22     dataPoints->push_back(new DataPoint(23, 35));
23     dataPoints->push_back(new DataPoint(35, 18));
24     dataPoints->push_back(new DataPoint(14, 3));
25     dataPoints->push_back(new DataPoint(17, 6));
26     dataPoints->push_back(new DataPoint(38, 15));
27     dataPoints->push_back(new DataPoint(26, 41));
28     dataPoints->push_back(new DataPoint(27, 38));
29     dataPoints->push_back(new DataPoint(30, 35));
30     dataPoints->push_back(new DataPoint(19, 12));
31     dataPoints->push_back(new DataPoint(24, 32));
32     dataPoints->push_back(new DataPoint(41, 26));
33     dataPoints->push_back(new DataPoint(38, 24));
34     dataPoints->push_back(new DataPoint(36, 21));
35     dataPoints->push_back(new DataPoint(30, 32));
36     dataPoints->push_back(new DataPoint(17, 3));
37     dataPoints->push_back(new DataPoint(12, 5));
38     dataPoints->push_back(new DataPoint(24, 38));
39     dataPoints->push_back(new DataPoint(14, 6));
40     dataPoints->push_back(new DataPoint(27, 32));
41     dataPoints->push_back(new DataPoint(17, 9));
42     dataPoints->push_back(new DataPoint(5, 40));
43 }
```

(x, y)
coordinates
of each data
point

This point is a **data outlier** which we will use later

View Lab 2 – main.cpp

```
void InitClusters()
{
    clusters = new vector<Cluster*>();
    for (int i{}; i < num clusters; ++i)
        clusters->push_back(new Cluster(i));
    // Assign each data point to an initial
    // cluster in a round-robin fashion
    for (size_t i{}; i < dataPoints->size(); ++i)
    {
        Cluster* c = clusters->at(i % clusters->size());
        dataPoints->at(i)->c = c;
        c->population++;
    }
}
```

Datapoints are assigned to an initial cluster in a **round-robin** fashion

View Lab 2 – main.cpp

```
int main()
{
    SimpleScreen ss(draw, eventHandler);
    ss.SetWorldRect(-5, -5, 45, 45);

    cout << "Press S to single step th
    cout << "Press Q to quit the appli

    InitDataPoints();

    InitClusters();

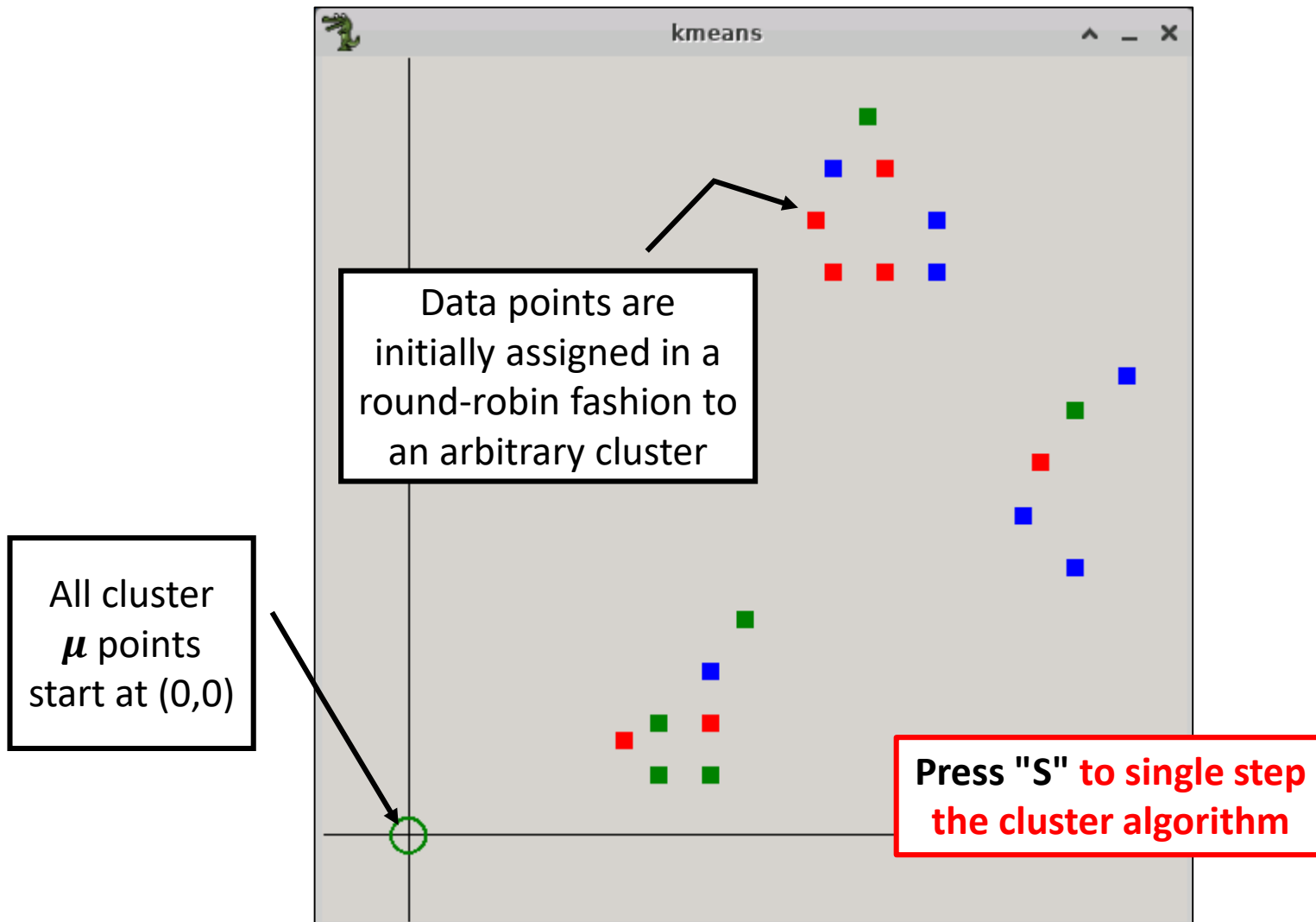
    ss.HandleEvents();

    return 0;
}
```

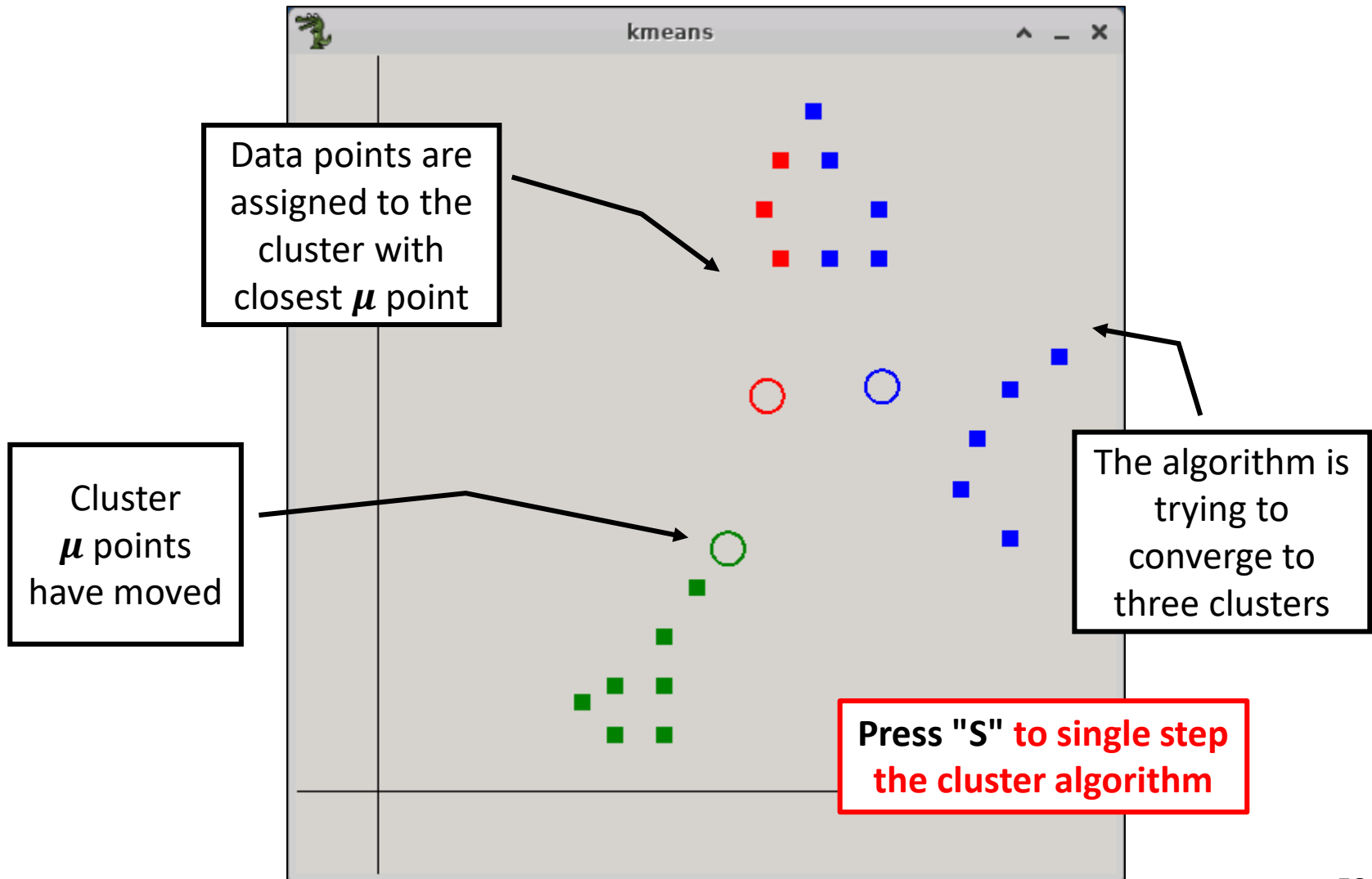
```
void draw(SimpleScreen& ss)
{
    ss.Clear();
    ss.DrawAxes();
    // Draw each data point
    for (auto dp : *dataPoints)
        ss.DrawRectangle(dp->c->clr, dp->x, dp->y, 1, 1, 0, true);
    // Draw each cluster's geometric center (the "mean")
    for (auto c : *clusters)
        ss.DrawCircle(c->x, c->y, 1, c->clr, 2);
    ss.Update();
}
```

```
void eventHandler(SimpleScreen& ss, ALLEGRO_EVENT& ev)
{
    if (ev.type == ALLEGRO_EVENT_KEY_CHAR)
    {
        switch (ev.keyboard.keycode)
        {
            case ALLEGRO_KEY_S:
                UpdateClusters();
                ss.Redraw();
                if (converged)
                    cout << "Cluster has converged" << endl;
                break;
            case ALLEGRO_KEY_Q:
                ss.Exit();
                break;
        }
    }
}
```

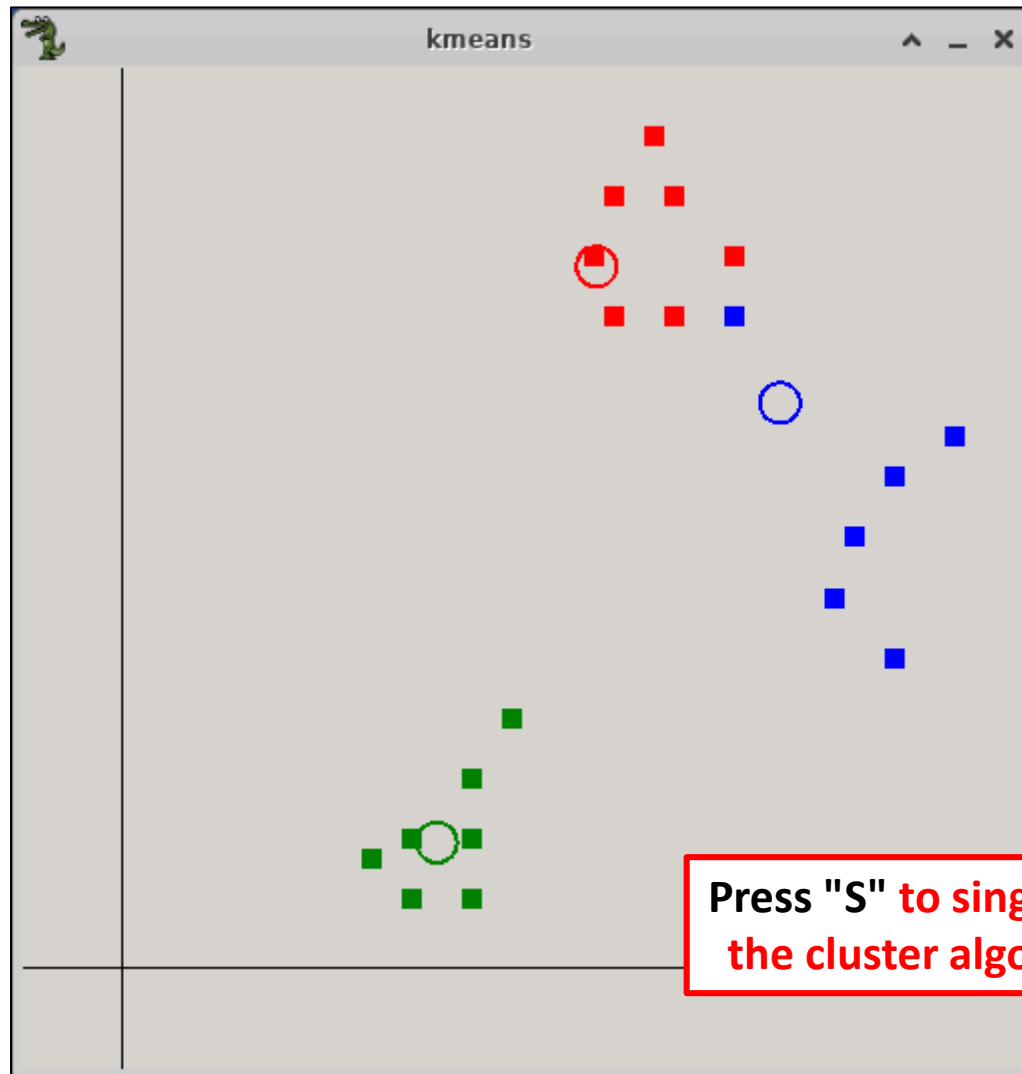

Run Lab 2 - Round 0



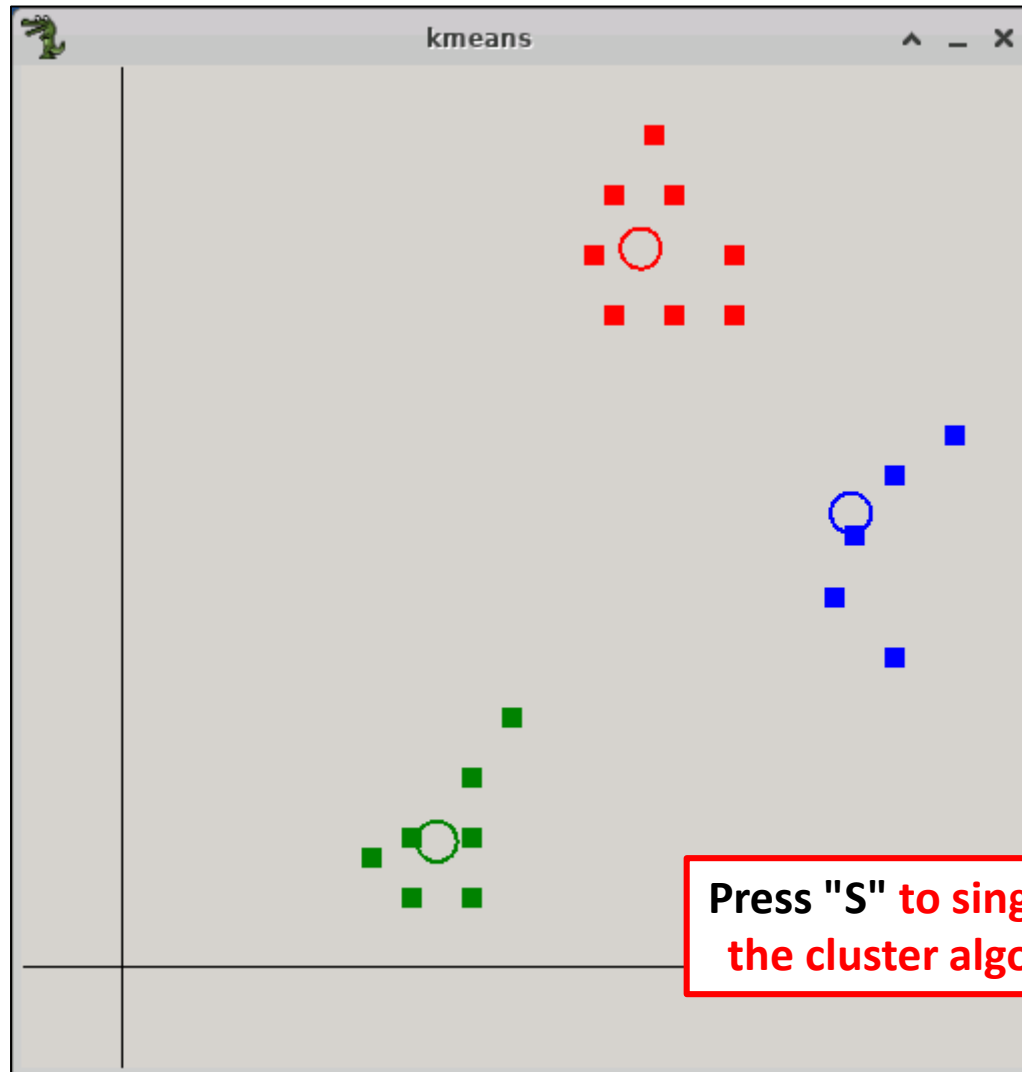
Check Lab 2 - Round 1



Check Lab 2 - Round 2

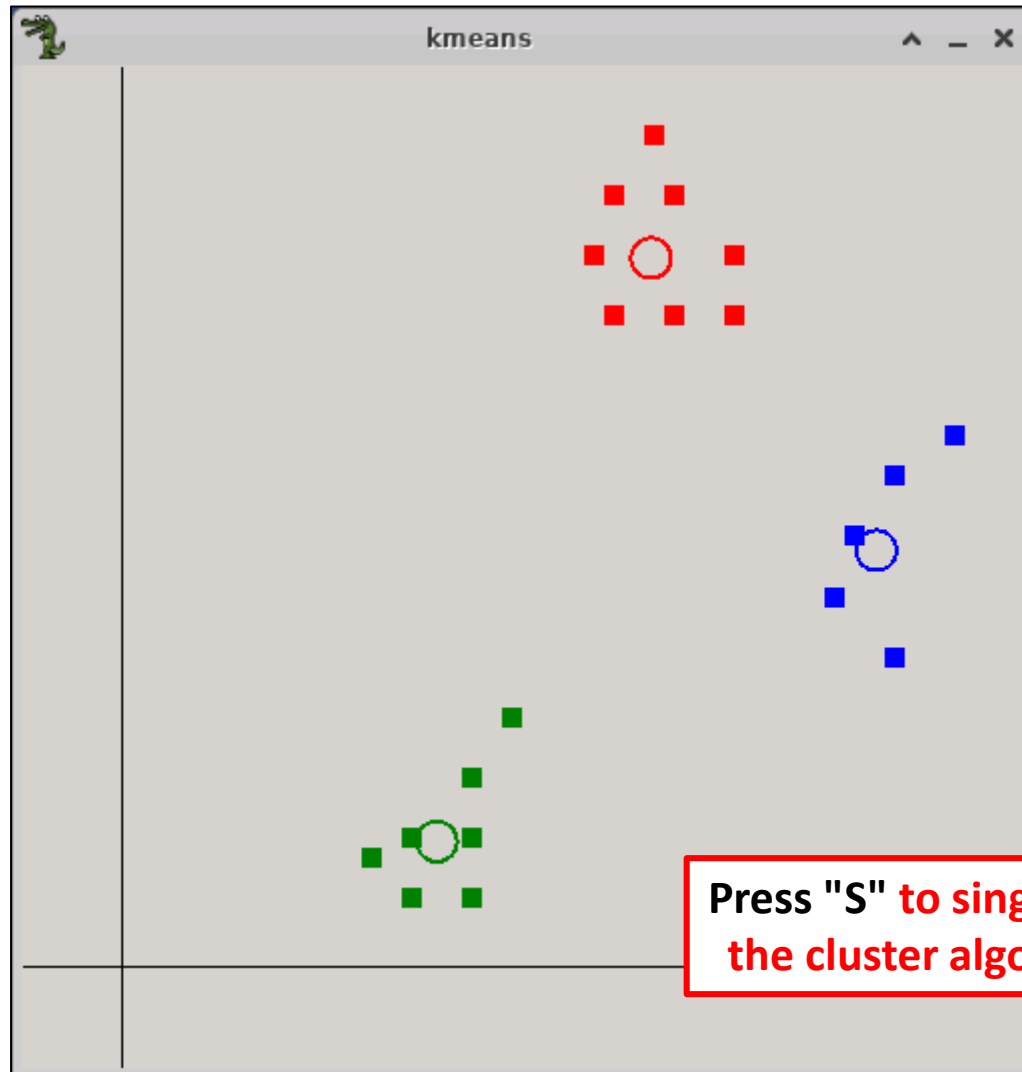


Check Lab 2 - Round 3

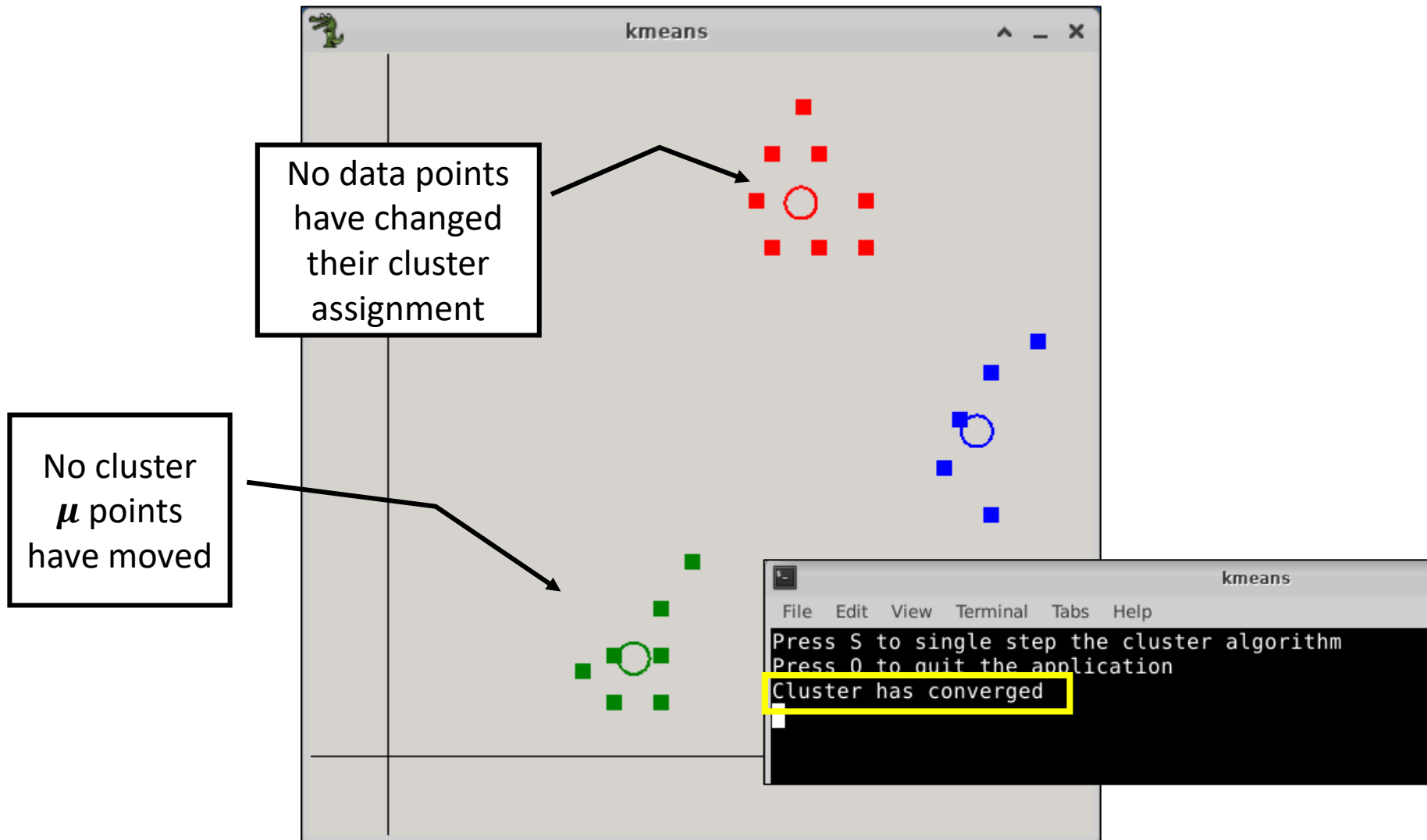


**Press "S" to single step
the cluster algorithm**

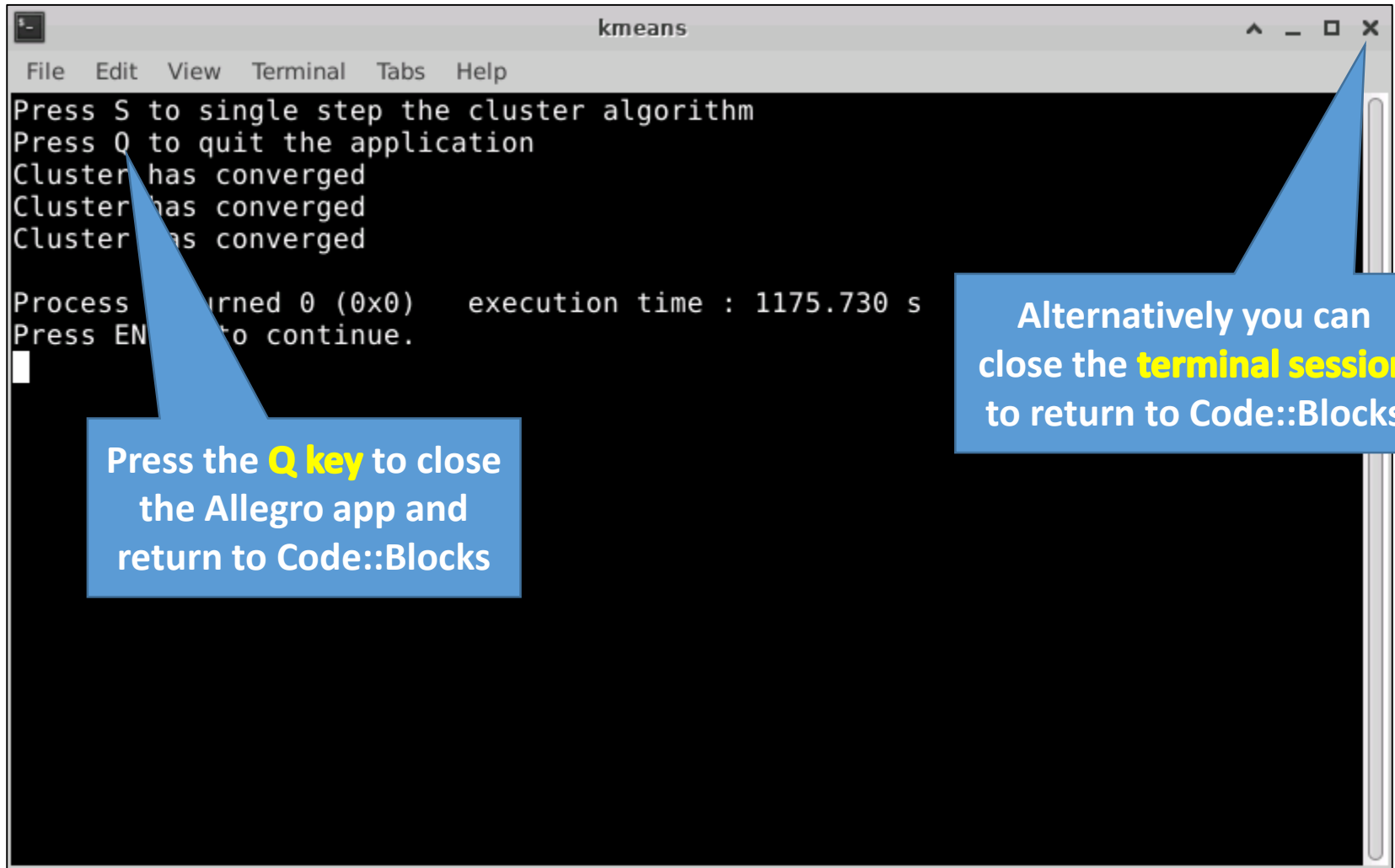
Check Lab 2 - Round 4



Check Lab 2 - Round 5



Quit Lab 2 - Round 5



```
kmeans
File Edit View Terminal Tabs Help
Press S to single step the cluster algorithm
Press Q to quit the application
Cluster has converged
Cluster has converged
Cluster has converged

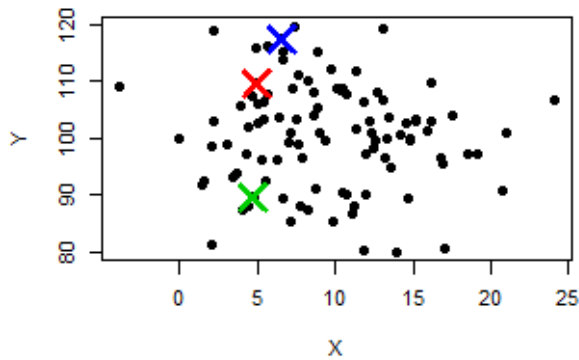
Process returned 0 (0x0)   execution time : 1175.730 s
Press ENTER to continue.
█
```

Press the **Q** key to close the Allegro app and return to Code::Blocks

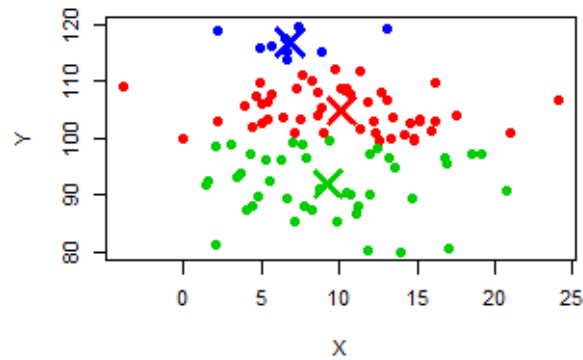
Alternatively you can close the **terminal session** to return to Code::Blocks

k-means Clustering

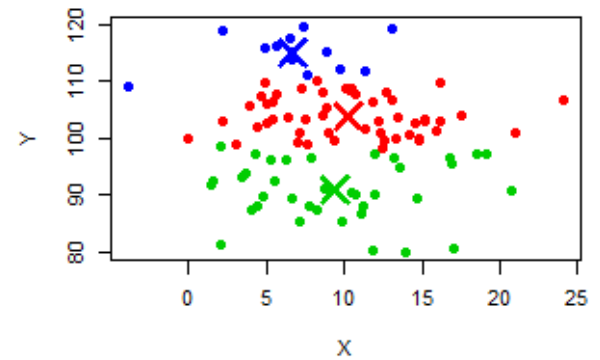
Iteration 1



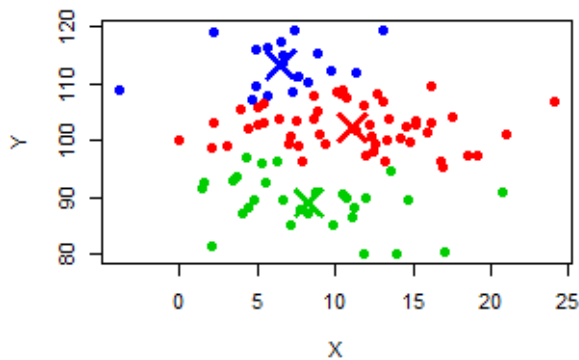
Iteration 2



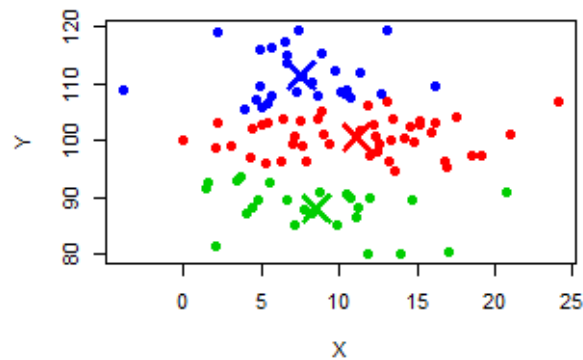
Iteration 3



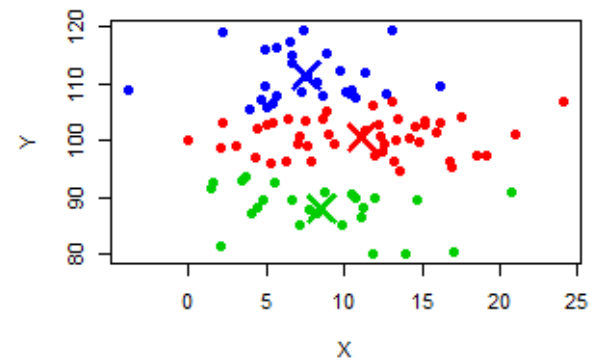
Iteration 6



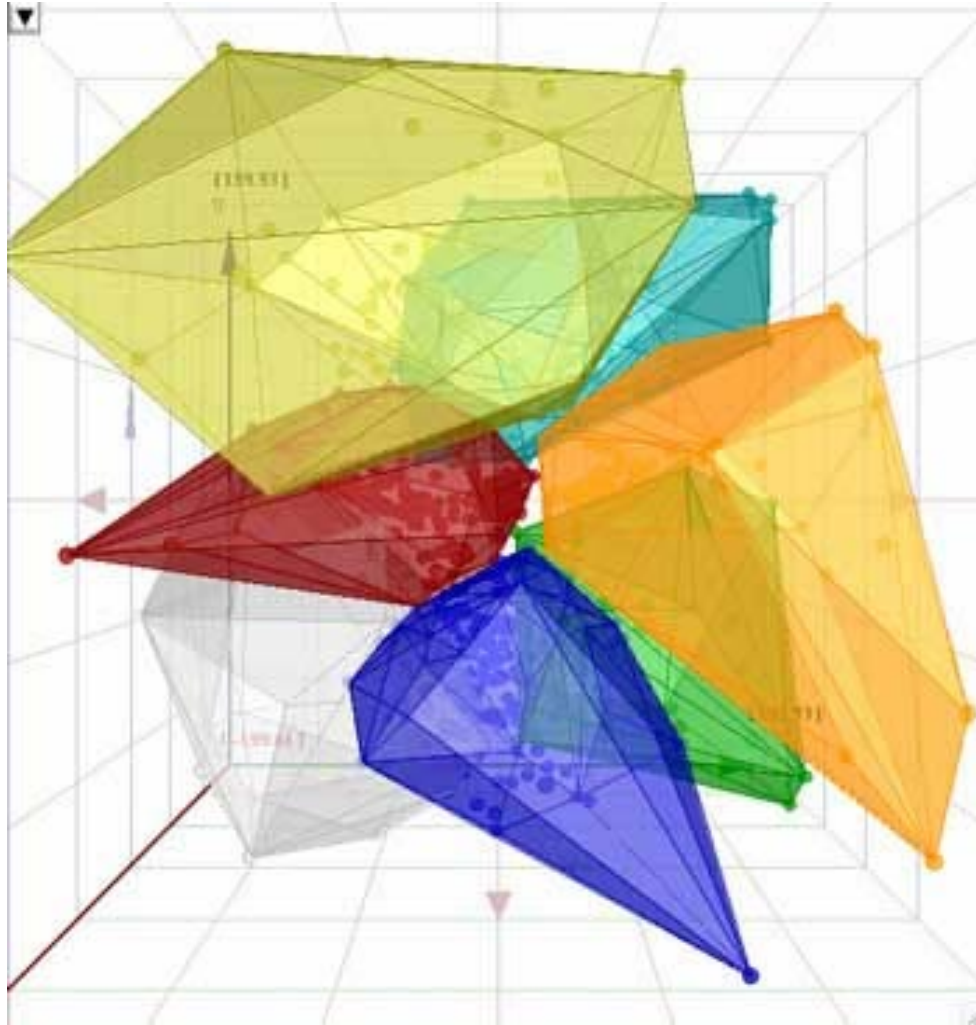
Iteration 9



Converged!



n-Dimensional k-means Clustering



View Lab 2 – main.cpp

```
void UpdateClusters()
{
    // Phase I: Calculate the new geometric mean of each
    // cluster based upon current data point assignments
    bool phase1_change{false};
    for (auto c : *clusters)
    {
        DataPoint newCenter;
        double count{};
        for (auto dp : *dataPoints)
        {
            if (dp->c == c)
            {
                newCenter.x += dp->x;
                newCenter.y += dp->y;
                count++;
            }
        }
        newCenter.x /= count;
        newCenter.y /= count;
        // Move cluster center (mean) if necessary
        if (newCenter.x != c->x || newCenter.y != c->y)
        {
            c->x = newCenter.x;
            c->y = newCenter.y;
            phase1_change = true;
        }
    }
}
```

```
// Phase II: Assign data points to nearest cluster
bool phase2_change{false};
for (auto dp : *dataPoints)
{
    double distMin = numeric_limits<double>::max();
    Cluster* cNearest = nullptr;
    for (auto c : *clusters)
    {
        double dist = GetDistance(dp->x, dp->y, c->x, c->y);
        if (dist < distMin)
        {
            distMin = dist;
            cNearest = c;
        }
    }
    // If the nearest cluster to this point is NOT
    // the point's currently assigned cluster...
    if (cNearest != dp->c)
    {
        // Then reassign data point to the new cluster, but only
        // if this is not the last point the current cluster.
        // Reassignment must never result in an empty cluster
        if (dp->c->population > 1)
        {
            dp->c->population--;
            // Assign point to new (nearer) cluster
            dp->c = cNearest;
            dp->c->population++;
            phase2_change = true;
        }
    }
}

if (mean_multiple == 0 && !phase1_change && !phase2_change && !converged)
{
    converged = true;
}
```

Edit Lab 2 – main.cpp

```
main.cpp [x]
1  #include "stdafx.h"
2  #include "simplescreen.h"
3  #include "kMeansClustering.h"
4
5  using namespace std;
6
7  int num_clusters{4};
8  double mean_multiple{0};
9
10 vector<DataPoint*> dataPoints;
11 vector<Cluster*> clusters{};
12 bool converged;
```

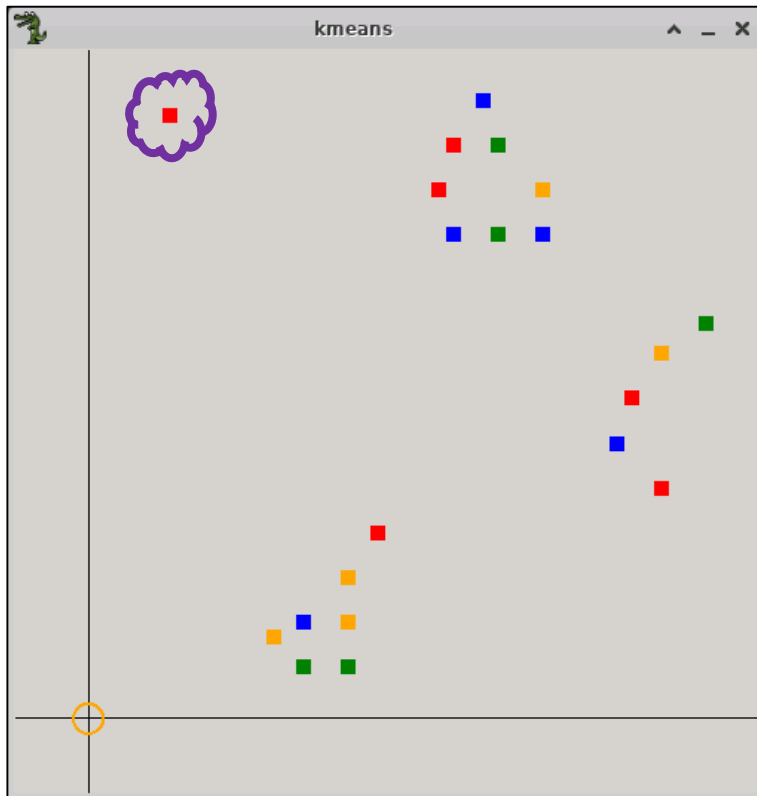
Increase
num_clusters to 4

Uncomment line 42 to
include a **data outlier**

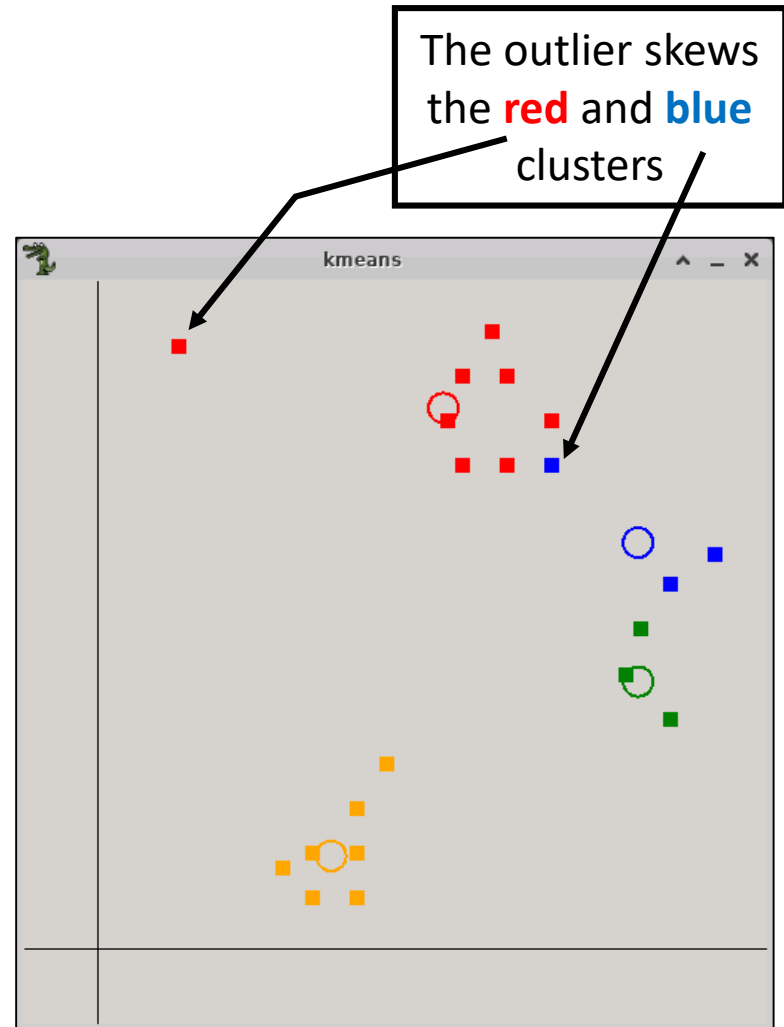
```
main.cpp [x]
19 void InitDataPoints()
20 {
21     dataPoints = new vector<DataPoint*>();
22     dataPoints->push_back(new DataPoint(23, 35));
23     dataPoints->push_back(new DataPoint(35, 18));
24     dataPoints->push_back(new DataPoint(14, 3));
25     dataPoints->push_back(new DataPoint(17, 6));
26     dataPoints->push_back(new DataPoint(38, 15));
27     dataPoints->push_back(new DataPoint(26, 41));
28     dataPoints->push_back(new DataPoint(27, 38));
29     dataPoints->push_back(new DataPoint(30, 35));
30     dataPoints->push_back(new DataPoint(19, 12));
31     dataPoints->push_back(new DataPoint(24, 32));
32     dataPoints->push_back(new DataPoint(41, 26));
33     dataPoints->push_back(new DataPoint(38, 24));
34     dataPoints->push_back(new DataPoint(36, 21));
35     dataPoints->push_back(new DataPoint(30, 32));
36     dataPoints->push_back(new DataPoint(17, 3));
37     dataPoints->push_back(new DataPoint(12, 5));
38     dataPoints->push_back(new DataPoint(24, 38));
39     dataPoints->push_back(new DataPoint(14, 6));
40     dataPoints->push_back(new DataPoint(27, 32));
41     dataPoints->push_back(new DataPoint(17, 9));
42     dataPoints->push_back(new DataPoint(5, 40));
43 }
```

Run Lab 2 – Data Outlier

Press "S" to single step
the cluster algorithm



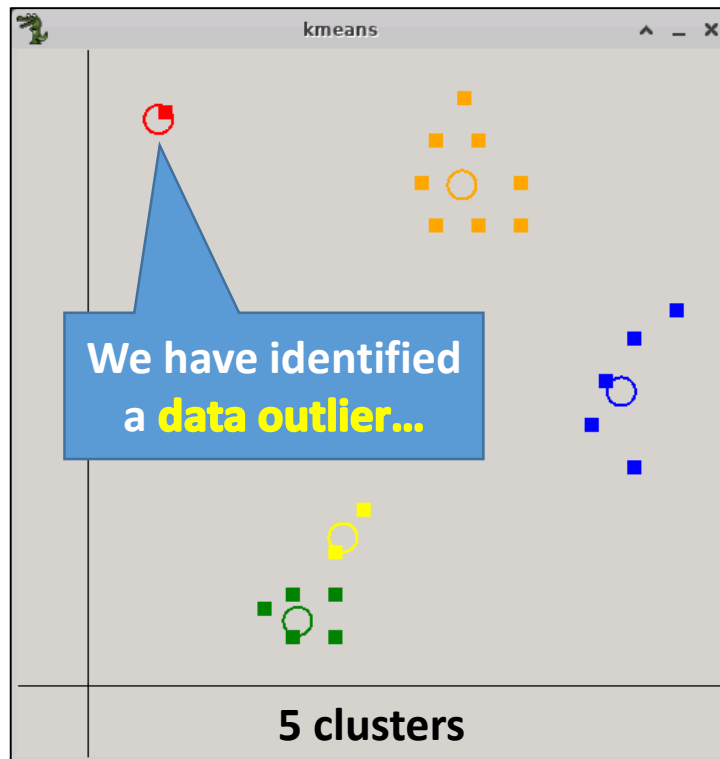
Round 0



Round 3

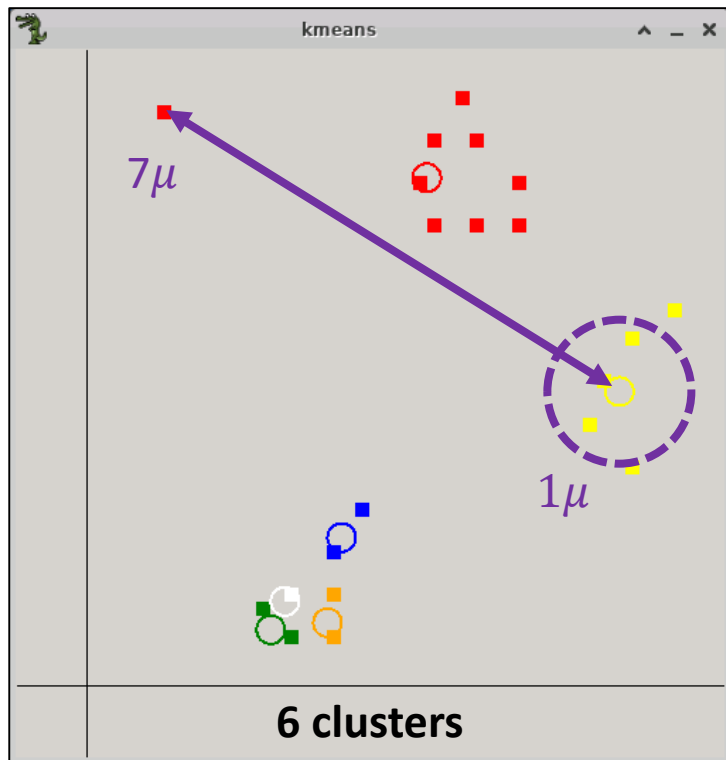
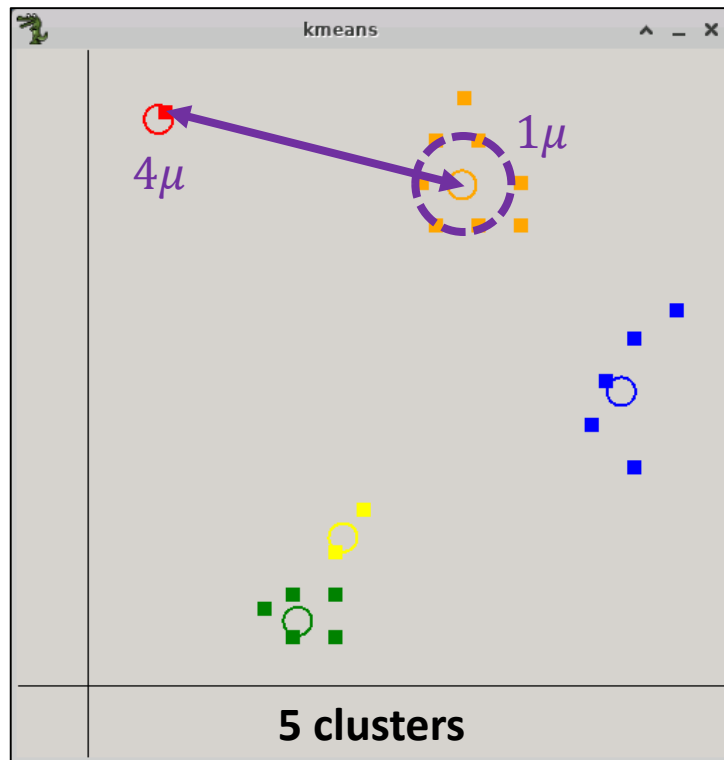
Edit Lab 2 – Research Questions

- Compare convergence with **num_clusters** set to 5 versus 6
- Is using more clusters a key to identifying data outliers?



Identifying Data Outliers

- Could a **mean (μ) distance cut-off** be used to gauge if a datapoint *reasonably* belongs to a given cluster?



Identifying Data Outliers

- Once the algorithm converges, possible data outliers can be identified by **evicting** any points that are too far away from the center of their assigned cluster
 - Reasonable data points should fall around the cluster's center point in a **normal** distribution. This means that **99.97%** of data points should be less than **3σ** away from the center of their cluster
 - Upon convergence, if the distance between a data point and the center of its cluster is greater than some **multiplier** of that cluster's mean (μ) spread, it probably is an outlier and should be evicted
 - If any data point is evicted, the algorithm must continue to iterate
- If a data point ultimately gets evicted from **all** clusters, it is probably a **data capture error** or a **statistical anomaly**
- This technique will be explored in Exam #3

Identifying Data Outliers

A Fast Outlier Detection Method for Big Data

Boyuan Liu, Wenhui Fan

State CIMS Engineering Research Center
Liuboyuan@

September 30, 2014

Big Data Outlier Detection, for Fun and Profit

Alex Woodie



As we discussed in the first part of this series, how you handle data outliers can determine whether your

flames out in
what to do with
ct them. That is

ngs at different
lenging to deal
e hand, the
golden buying
tliers may just be
goose chase.

Abstra
tion sy
tion m
propos
the nev
measu
of rapi
results
out ob

Keywo

NATO Science for Peace and Security Series
D: Information and Communication Security - Vol. 19

Mining Massive Data Sets for Security

Advances in Data Mining, Search, Social
Networks and Text Mining, and their Applications
to Security

Now you know...

- We can use **k-means clustering** to categorize data points by its nearness to a similar group's statistical means
 - Data sets which are homogenous (have a more uniform distribution of values) **may not converge** to a meaningful set of clusters since the data lacks *sharpness*
 - We can use k-means **to identify data outliers** so we can avoid allowing those anomalies from skewing the main observations – this process is called evicting a point from the cluster
- We can extend k-means clustering to analyze **multi-dimensional data sets** to look for trends
 - This is exactly how Amazon and Netflix provide you purchase recommendations based upon your shared interests & tastes compared with other buyers – a **very effective** selling technique!