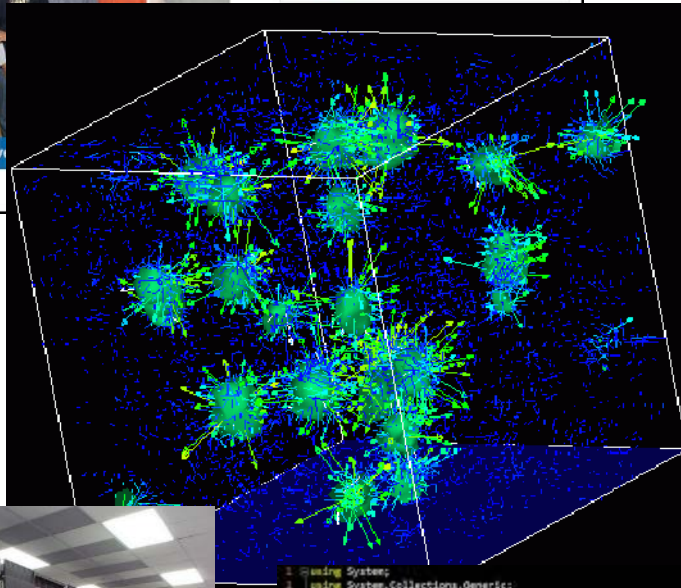




Survey of Scientific Computing (SciComp 301)

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 08
Algorithms,
Series Convergence

Session Goals

- Generate & sum pseudo-infinite series in code
 - Map sigma \sum elements to code constructs such as **for()** loops and *accumulating* variables
 - Compare **convergent** vs. **divergent** series
- Understand the **Basel Problem** and Euler's amazing result
- Learn how Euclid's Greatest Common Divisor (**GCD**) algorithm provides the correct result without factoring
- Determine the probability that two random positive integers are **coprime** (share no common factors other than 1)
- Write code to explore the **birthday paradox**

The Basel Problem

- Calculate the sum of the reciprocals of the positive integers in *batches* of **1,000** terms (up to **10,000** terms)

$$\textit{Sum} = \sum_{n=1}^{\infty} \frac{1}{n} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \dots$$

- This is called the **harmonic series**
- Does this series **converge** to a single value – or does it **diverge** (grow without bounds)?
- Find the value of $\sqrt{\textit{Sum} * 6}$

Open Lab 1 – Basel Problem

```
basel-series.cpp ✕
1 // basel-series.cpp
2
3 #include "stdafx.h"
4
5 using namespace std;
6
7 int main()
8 {
9     double sum;
10
11     cout.imbue(std::locale(""));
12
13     for (int limit{ 1000 }; limit <= 10000; limit += 1000) {
14         sum = 0;
15         for (int n{ 1 }; n < limit; ++n)
16             sum += 1.0 / n;
17
18         cout << "Sum of reciprocals of positive integers <= ";
19         cout << setw(6) << limit << " = ";
20         cout << setprecision(14) << sum << endl;
21     }
22
23     cout << endl << "Magic Number = "
24         << sqrt(sum * 6) << endl;
25
26     return 0;
27 }
28
```

main()

- cout.imbue()
- Using {} to init vars
- Prefix ++ op
- Explicit **1.0**
- setw()
- setprecision()
- sqrt()

Run Lab 1 – Basel Problem

```
basel-series
File Edit View Terminal Tabs Help
Sum of reciprocals of positive integers <= 1,000 = 7.4844708605503
Sum of reciprocals of positive integers <= 2,000 = 8.1778681036103
Sum of reciprocals of positive integers <= 3,000 = 8.5834165566258
Sum of reciprocals of positive integers <= 4,000 = 8.8711402997952
Sum of reciprocals of positive integers <= 5,000 = 9.0943088529844
Sum of reciprocals of positive integers <= 6,000 = 9.2766470774636
Sum of reciprocals of positive integers <= 7,000 = 9.4308096626668
Sum of reciprocals of positive integers <= 8,000 = 9.5643499842614
Sum of reciprocals of positive integers <= 9,000 = 9.6821399646355
Sum of reciprocals of positive integers <= 10,000 = 9.7875060360443

Magic Number = 7.663226227658

Process returned 0 (0x0)   execution time : 0.012 s
Press ENTER to continue.
```

The harmonic series diverges

$$\begin{aligned}\sum_{k=1}^{\infty} \frac{1}{k} &= 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \dots \\ &= 1 + \left(\frac{1}{2}\right) + \left(\frac{1}{3} + \frac{1}{4}\right) + \left(\frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8}\right) + \left(\frac{1}{9} + \dots\right) \\ &\geq 1 + \left(\frac{1}{2}\right) + \left(\frac{1}{4} + \frac{1}{4}\right) + \left(\frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8}\right) + \left(\frac{1}{16} + \dots\right) \\ &= 1 + \left(\frac{1}{2}\right) + \left(\frac{1}{2}\right) + \left(\frac{1}{2}\right) + \dots = \infty.\end{aligned}$$

Basel problem

From Wikipedia, the free encyclopedia

The **Basel problem** is a problem in mathematical analysis with relevance to number theory, first posed by Pietro Mengoli in 1644 and solved by Leonhard Euler in 1734^[1] and read on 5 December 1735 in *The Saint Petersburg Academy of Sciences* (Russian: Петербургская Академия наук).^[2] Since the problem had withstood the attacks of the leading mathematicians of the day, Euler's solution brought him immediate fame when he was twenty-eight. Euler generalised the problem considerably, and his ideas were taken up years later by Bernhard Riemann in his seminal 1859 paper *On the Number of Primes Less Than a Given Magnitude*, in which he defined his zeta function and proved its basic properties. The problem is named after Basel, hometown of Euler as well as of the Bernoulli family who unsuccessfully attacked the problem.

The Basel problem asks for the precise summation of the reciprocals of the squares of the natural numbers, i.e. the precise sum of the infinite series:

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \lim_{n \rightarrow \infty} \left(\frac{1}{1^2} + \frac{1}{2^2} + \cdots + \frac{1}{n^2} \right).$$

The sum of the series is approximately equal to 1.644934 ⁹⁵A013661. The Basel problem asks for the exact sum of this series (in closed form), as well as a proof that this sum is correct.

Edit Lab 1 – Basel Problem

- Modify the program to calculate the sum of **the squares** of the reciprocals of the positive integers

$$\text{Sum} = \sum_{n=1}^{\infty} \frac{1}{n^2} = 1 + \frac{1}{4} + \frac{1}{9} + \frac{1}{16} + \frac{1}{25} + \frac{1}{36} + \frac{1}{49} + \dots$$

- Does this sequence converge to a single value – or does it diverge (grow without bounds)? If it converges, what is its **exact** value?
- Don't forget we are now **squaring n** in the *denominator*, so update the code accordingly (edit line #16)
- After this change does the $\sqrt{\text{Sum}} * 6$ look more familiar?

Run Lab 1 – Basel Problem

```
basel-series
File Edit View Terminal Tabs Help
Sum of reciprocals of positive integers <= 1,000 = 1.6439335666816
Sum of reciprocals of positive integers <= 2,000 = 1.6444339418274
Sum of reciprocals of positive integers <= 3,000 = 1.6446006779532
Sum of reciprocals of positive integers <= 4,000 = 1.6446840355956
Sum of reciprocals of positive integers <= 5,000 = 1.6447340468469
Sum of reciprocals of positive integers <= 6,000 = 1.6447673862919
Sum of reciprocals of positive integers <= 7,000 = 1.6447911995008
Sum of reciprocals of positive integers <= 8,000 = 1.6448090590354
Sum of reciprocals of positive integers <= 9,000 = 1.6448229495641
Sum of reciprocals of positive integers <= 10,000 = 1.6448340618481

Magic Number = 3.1414971543976

Process returned 0 (0x0)   execution time
Press ENTER to continue.
```

The series converges

$$\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$$

$$\sum_{k=1}^{\infty} \frac{1}{k^4} = \frac{\pi^4}{90}$$

$$\sum_{k=1}^{\infty} \frac{1}{k^6} = \frac{\pi^6}{945}$$

$$\sum_{k=1}^{\infty} \frac{1}{k^{26}} = \frac{2^{24} \times 76977927 \pi^{26}}{27!}$$



Greatest Common Divisor (GCD)

Example: **What is the GCD of 231 and 182?** In step 0, **A** is always greater than or equal to **B**. In steps 1 and beyond, the **A** value is the *greater* of the prior step's **B** or (**A**-**B**) values. The **B** value is the *lesser* of either the prior step's **B** or (**A** - **B**) values. The algorithm stops when **A** - **B** = 0, and the GCD was the very last **B** value. Follow along with each step in the table below:

Finding the GCD of 231 and 182			
Step	A	B	A - B
0	231	182	49
1	182	49	133
2	133	49	84
3	84	49	35
4	49	35	14
5	35	14	21
6	21	14	7
7	14	7	7
8	7	7	0

What divides A and B must also divide the ***difference*** of A - B

Why?

Given $\{A, B, a, b, r\} \in \mathbb{Z}$

$$A = a * r, B = b * r$$

$$(A - B) = a * r - b * r$$

$$a - b = \frac{(A - B)}{r}$$

Open Lab 2 – Euclid's GCD

```
euclid-gcd.cpp ✕
1 // euclid-gcd.cpp
2
3 #include "stdafx.h"
4
5 using namespace std;
6
7 int GCD(int a, int b)
8 {
9     // Implement the GCD algorithm
10
11     return b;
12 }
13
14 int main()
15 {
16     int a = 231;
17     int b = 182;
18
19     cout << "The GCD of " << a
20          << " and " << b << " = " <<
21          GCD(a,b) << endl;
22
23     return 0;
24 }
25
```



Finding the GCD of 231 and 182			
Step	A	B	A - B
0	231	182	49
1	182	49	133
2	133	49	84
3	84	49	35
4	49	35	14
5	35	14	21
6	21	14	7
7	14	7	7
8	7	7	0

Ensure $a \geq b$

Edit Lab 2 – Euclid's GCD

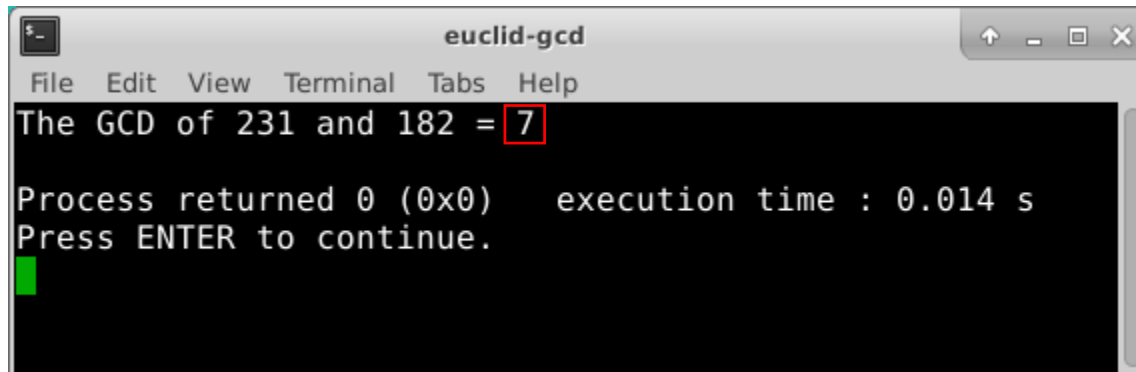
```
7 int GCD(int a, int b)
8 {
9     if (a < b)
10        swap(a, b);
11
12     int c = a - b;
13
14     while (c > 0) {
15         if (c > b)
16             a = c;
17         else {
18             a = b;
19             b = c;
20         }
21         c = a - b;
22     }
23
24     return b;
25 }
```

Finding the GCD of 231 and 182

Step	A	B	A - B
0	231	182	49
1	182	49	133
2	133	49	84
3	84	49	35
4	49	35	14
5	35	14	21
6	21	14	7
7	14	7	7
8	7	7	0

Run Lab 2 – Euclid's GCD

Finding the GCD of 231 and 182			
Step	A	B	A - B
0	231	182	49
1	182	49	133
2	133	49	84
3	84	49	35
4	49	35	14
5	35	14	21
6	21	14	7
7	14	7	7
8	7	7	0



```
euclid-gcd
File Edit View Terminal Tabs Help
The GCD of 231 and 182 = 7
Process returned 0 (0x0)   execution time : 0.014 s
Press ENTER to continue.
█
```

Coprime Probability

- Calculate the average number of times (the **probability**) a **million** pairs of random integers ($1 \leq n \leq 100,000$) are coprime (their **GCD == 1**)
- What does this experiment estimate to be the **odds that two randomly chosen integers will share no common factors?**
- Find the value of $\sqrt{\frac{6}{\text{probability}}}$

Open Lab 3 – Coprime Probability

```
coprime-probability.cpp ✕
1 // coprime-probability.cpp
2
3 #include "stdafx.h"
4
5 using namespace std;
6
7 int GCD(int a, int b)
8 {
9     return b == 0 ? a : GCD(b, a % b);
10 }
11
12 int main()
13 {
14     seed_seq seed{ 2016 };
15     default_random_engine generator{ seed };
16     uniform_int_distribution<int> distribution(1, 100000);
17
18     double maxIterations{ 1000000 };
19     double coprimePairs{};
20
21     for (double i{}; i < maxIterations; ++i) {
22         int a = distribution(generator);
23         int b = distribution(generator);
24         if (GCD(a, b) == 1)
25             coprimePairs++;
26     }
27
28     double coprimeProbability = coprimePairs / maxIterations;
29
30     cout << "Probability two random integers are coprime = "
31          << setprecision(14) << coprimeProbability << endl;
32
33     cout << endl << "Hidden constant = "
34          << sqrt(6 / coprimeProbability) << endl;
35
36     return 0;
37 }
38
```

Run Lab 3 – Coprime Probability


```
coprime-probability
File Edit View Terminal Tabs Help
Probability two random integers are coprime = 0.608189
Hidden constant = 3.14091616496
Process returned 0 (0x0)   execution time : 0.378 s
Press ENTER to continue.
```

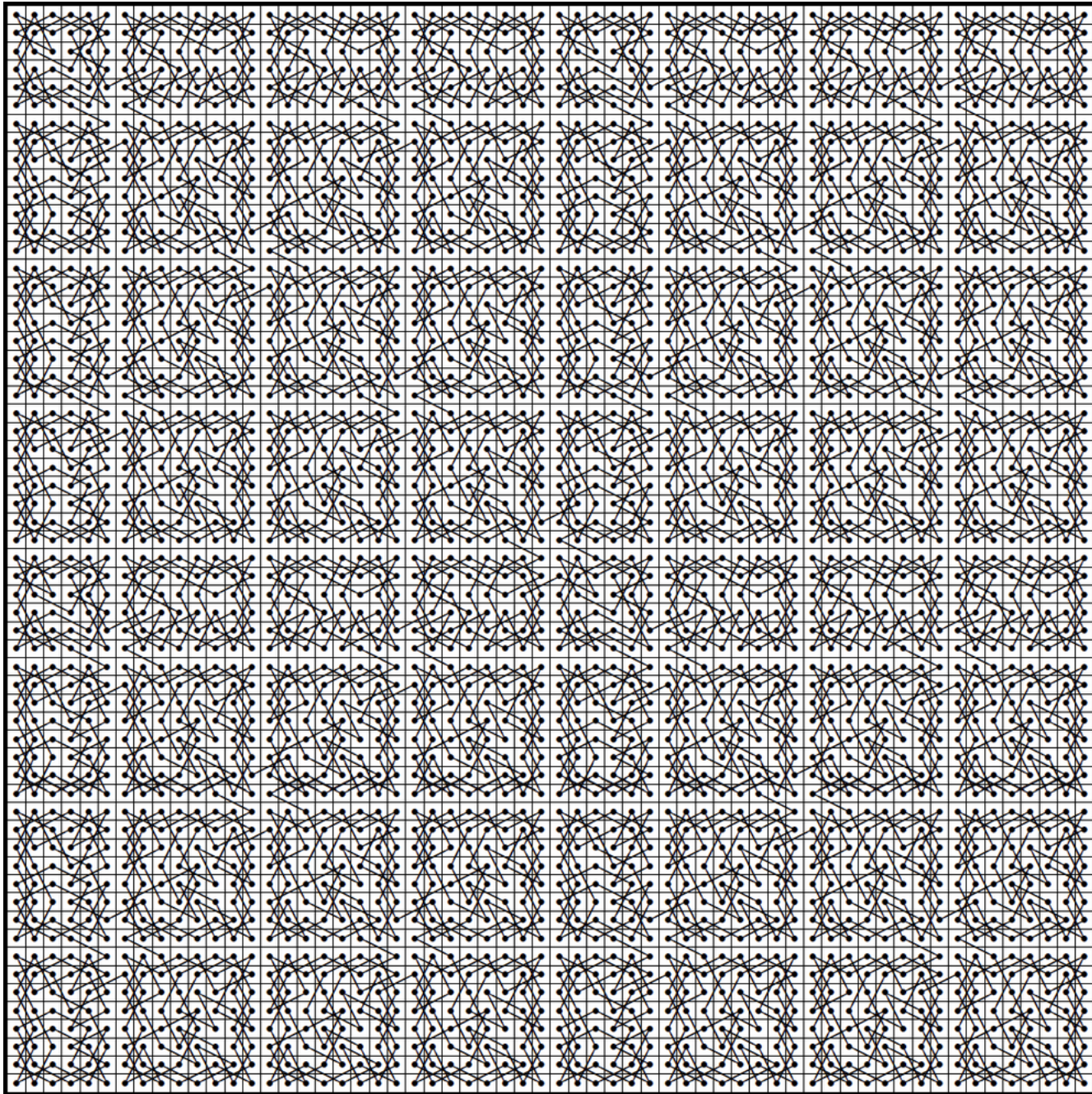
$$0.607927102 \approx 61\% \approx \frac{6}{\pi^2}$$

$$\frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \dots = \frac{\pi^2}{6}$$



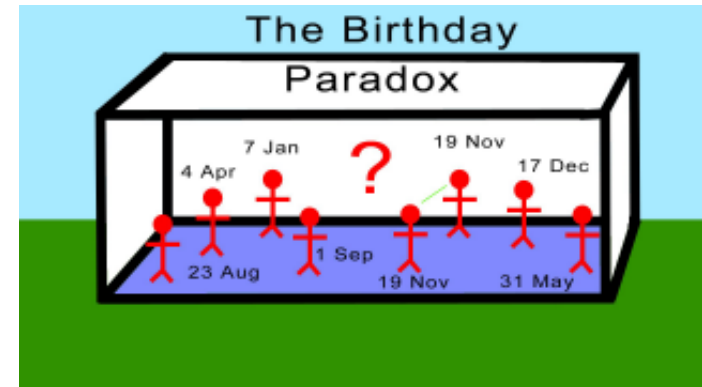
Euler's Knight Tour

35	40	47	44	61	08	15	12
46	43	36	41	14	11	62	09
39	34	45	48	07	60	13	16
50	55	42	37	22	17	10	63
33	38	49	54	59	06	23	18
56	51	28	31	26	21		03
29	32	53	58	05	02	19	24
52	57	30	27	20	25	04	01



Birthday Paradox

- Given a class size of **n** students, write a program to calculate the *probability* that at least two students in that class share the same birthday
- Your code should calculate this probability for **10,000** classes, each between **2** and **80** students inclusive
- Assume there is only 365 days in a year (no leap years)
- What is the **minimum** required class size to have **> 50%** probability of two similar birthdays?




```

1 // BirthdayParadox.cpp
2
3 #include "stdafx.h"
4
5 using namespace std;
6
7 int main()
8 {
9     seed_seq seed{ 2016 };
10    default_random_engine generator{ seed };
11    uniform_int_distribution<int> distribution(0, 364);
12
13    for (int students = 2; students <= 80; students++)
14    {
15        int totalIterations = 10000;
16        int matchCount = 0;
17
18        vector<int>* birthdays = new vector<int>(students, 0);
19
20        for (int iteration = 0; iteration < totalIterations; iteration++) {
21            // Initialize the birthdays array with a random day between 0 and 364
22            for (int i = 0; i < birthdays->size(); i++)
23                birthdays->at(i) = distribution(generator);
24
25            // Compare birthdays of each person to the remaining people
26            // Note: Only loop until the first match is found
27            bool foundMatch = false;
28            // TODO: Insert your code here
29
30            if (foundMatch)
31                matchCount++;
32        }
33
34        delete birthdays;
35
36        cout << "Probability of matching birthdays among "
37             << setw(2) << students << " people = "
38             << fixed << setprecision(4) << double matchCount / totalIterations
39             << endl;
40    }
41
42    return 0;
43 }

```

Open Lab 4 Birthday Paradox

Add the code here to
find if any two students
have the same birthday



```

1 // BirthdayParadox.cpp
2
3 #include "stdafx.h"
4
5 using namespace std;
6
7 int main()
8 {
9     seed_seq seed{ 2016 };
10    default_random_engine generator{ seed };
11    uniform_int_distribution<int> distribution(0, 364);
12
13    for (int students = 2; students <= 80; students++)
14    {
15        int totalIterations = 10000;
16        int matchCount = 0;
17
18        vector<int>* birthdays = new vector<int>(students, 0);
19
20        for (int iteration = 0; iteration < totalIterations; iteration++) {
21            // Initialize the birthdays array with a random day between 0 and 364
22            for (int i = 0; i < birthdays->size(); i++)
23                birthdays->at(i) = distribution(generator);
24
25            // Compare birthdays of each person to the remaining people
26            // Note: Only loop until the first match is found
27            bool foundMatch = false;
28            for (int j{0}; !foundMatch && j < birthdays->size() - 1; j++)
29                for (int k{ j + 1 }; !foundMatch && k < birthdays->size(); k++)
30                    if (birthdays->at(j) == birthdays->at(k))
31                        foundMatch = true;
32
33            if (foundMatch)
34                matchCount++;
35        }
36
37        delete birthdays;
38
39        cout << "Probability of matching birthdays among "
40             << setw(2) << students << " people = "
41             << fixed << setprecision(4) << (double)matchCount / totalIterations
42             << endl;
43    }
44
45    return 0;
46 }

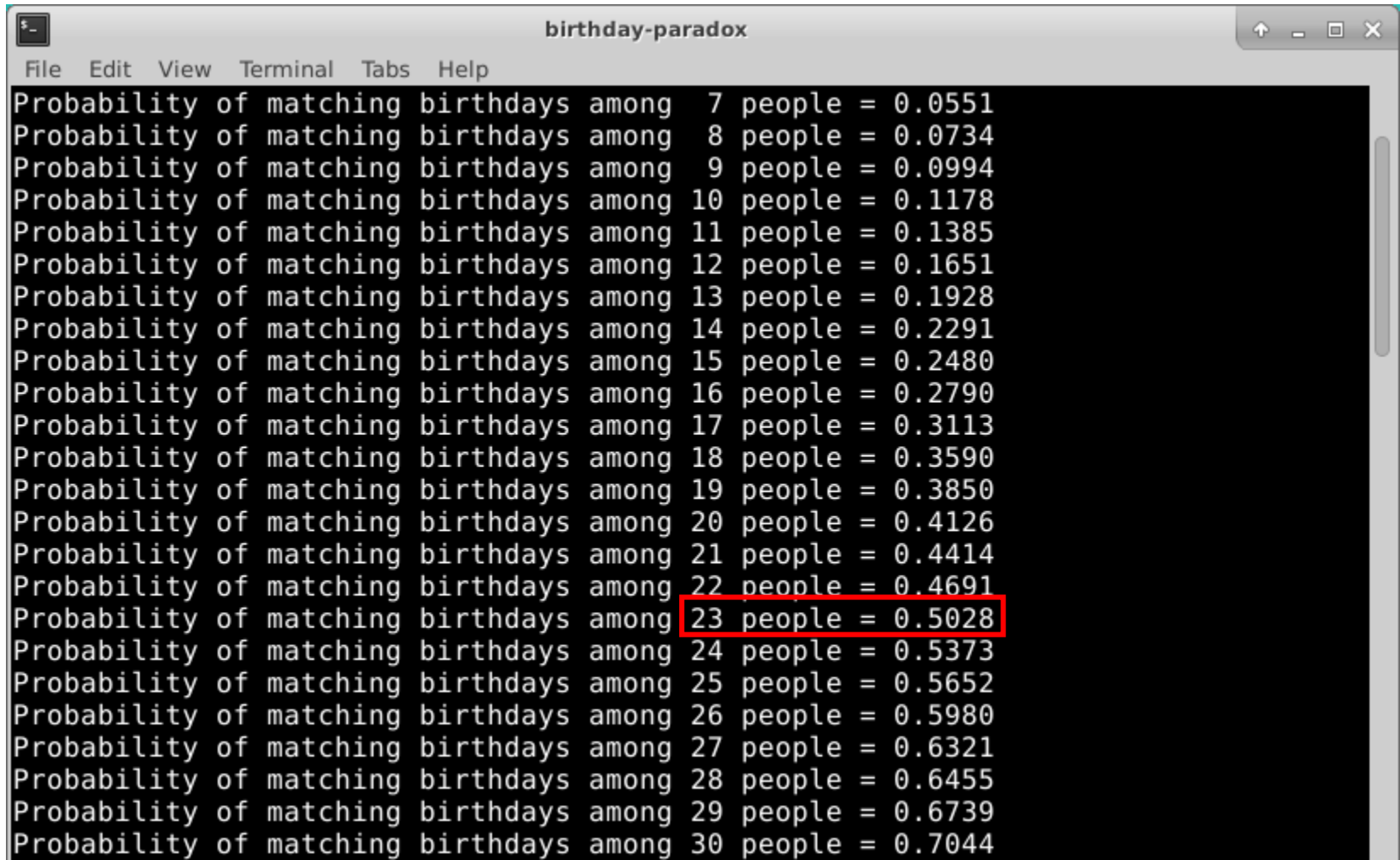
```

Edit

Lab 4

Birthday Paradox

Run Lab 4 – Birthday Paradox



```
birthday-paradox
File Edit View Terminal Tabs Help
Probability of matching birthdays among 7 people = 0.0551
Probability of matching birthdays among 8 people = 0.0734
Probability of matching birthdays among 9 people = 0.0994
Probability of matching birthdays among 10 people = 0.1178
Probability of matching birthdays among 11 people = 0.1385
Probability of matching birthdays among 12 people = 0.1651
Probability of matching birthdays among 13 people = 0.1928
Probability of matching birthdays among 14 people = 0.2291
Probability of matching birthdays among 15 people = 0.2480
Probability of matching birthdays among 16 people = 0.2790
Probability of matching birthdays among 17 people = 0.3113
Probability of matching birthdays among 18 people = 0.3590
Probability of matching birthdays among 19 people = 0.3850
Probability of matching birthdays among 20 people = 0.4126
Probability of matching birthdays among 21 people = 0.4414
Probability of matching birthdays among 22 people = 0.4691
Probability of matching birthdays among 23 people = 0.5028
Probability of matching birthdays among 24 people = 0.5373
Probability of matching birthdays among 25 people = 0.5652
Probability of matching birthdays among 26 people = 0.5980
Probability of matching birthdays among 27 people = 0.6321
Probability of matching birthdays among 28 people = 0.6455
Probability of matching birthdays among 29 people = 0.6739
Probability of matching birthdays among 30 people = 0.7044
```

Now you know...

- How to generate a pseudo-infinite series (harmonic and Basel) and map \sum elements to code constructs
- How the **GCD** works without having to factor a or b
- The closed-form (analytic) solution to the Basel series = $\frac{\pi^2}{6}$
- The probability two random integers are coprime = $\frac{6}{\pi^2}$
- Some statistics are only meaningful after generating a **very large** sample set – everything in scientific computing is **big**
- In a room of **80 people**, there is **~100%** chance that two people will share the same birthday!
- **Euler was smart!** 😊