Survey of Scientific Computing
(SciComp 301)

Dave Biersach
Brookhaven National Laboratory
dbiersach@bnl.gov

Session 08
Algorithms,
Series Convergence

Session Goals

- Generate & sum pseudo-infinite series in code

  - Map sigma ∑ 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

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

- Calculate the sum of the reciprocals of the positive integers up to a maximum of **10,000** terms

- Start by finding the sum of the first **1,000** terms, then the sum of the first **2,000** terms, then the first **3,000** terms, etc.

- 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 $\sqrt{Sum * 6}$

Open Lab 1 – Basel Problem

```cpp
#include "stdafx.h"

using namespace std;

int main()
{
    double sum;

    cout.imbue(std::locale(""));

    for (int limit{ 1000 }; limit <= 10000; limit += 1000) {
        sum = 0;
        for (int n{ 1 }; n < limit; ++n)
            sum += 1.0 / n;

        cout << "Sum of reciprocals of positive integers <= ";
        cout << setw(6) << limit << " = ";
        cout << setprecision(14) << sum << endl;
    }

    cout << endl << "Magic Number = "
        << sqrt(sum * 6) << endl;

    return 0;
}
```

main()

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

Run Lab 1 – Basel Problem

```
basel-series                                    ↑ _ □ X
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

$$\sum_{k=1}^{\infty} \frac{1}{k} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \ldots$$

$$= 1 + \left(\frac{1}{2}\right) + \left(\frac{1}{3} + \frac{1}{4}\right) + \left(\frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8}\right) + \left(\frac{1}{9} + \ldots\right.$$

$$\geq 1 + \left(\frac{1}{2}\right) + \left(\frac{1}{4} + \frac{1}{4}\right) + \left(\frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8}\right) + \left(\frac{1}{16} + \ldots\right.$$

$$= 1 + \left(\frac{1}{2}\right) + \left(\frac{1}{2}\right) + \left(\frac{1}{2}\right) + \ldots = \infty.$$

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:

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

The sum of the series is approximately equal to **1.644934** A013661. 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

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

- 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 **(hint: edit line #14)**

- After this change does the $\sqrt{\boldsymbol{Sum} * 6}$ 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 tim
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 $\{A, B, a, b, r\} \in \mathbb{Z}$

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

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

$$a - b = \frac{(A - B)}{r}$$
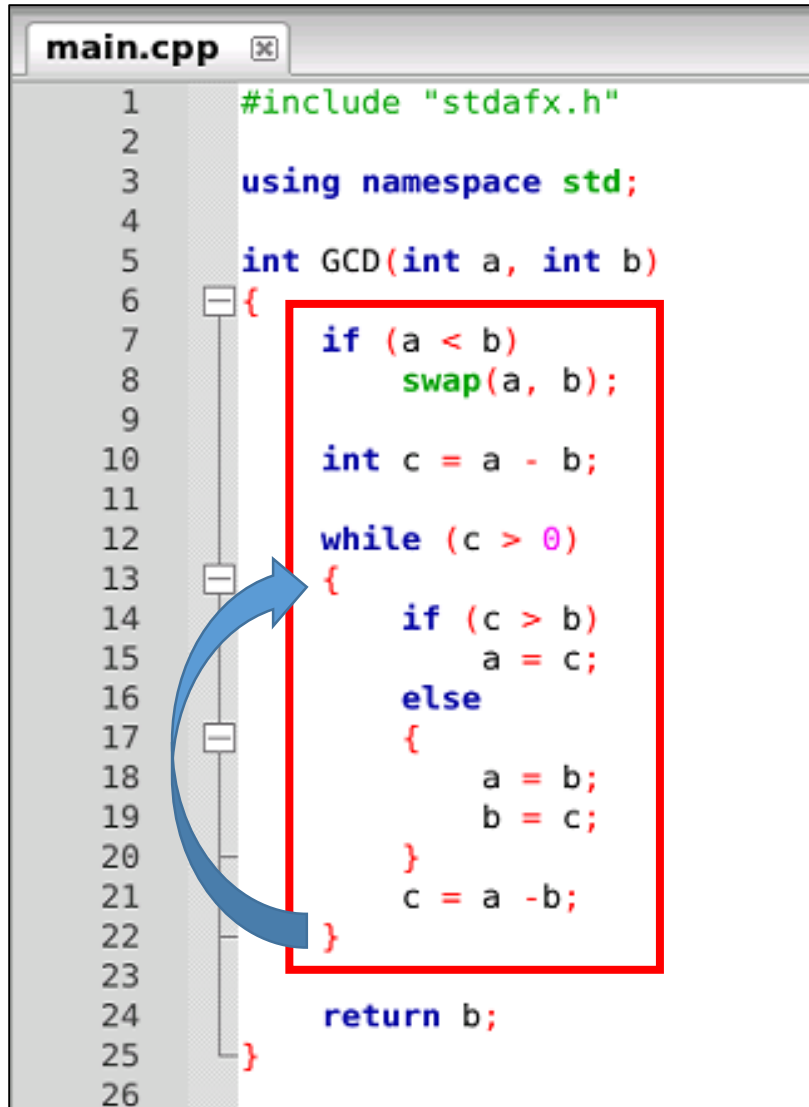
9

Open Lab 2 – Euclid's GCD

```cpp
#include "stdafx.h"

using namespace std;

int GCD(int a, int b)
{
    // Implement the GCD algorithm

    return b;
}

int main()
{
    int a = 231;
    int b = 182;

    cout << "The GCD of " << a
        << " and " << b << " = "<<
        GCD(a,b) << endl;

    return 0;
}
```

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

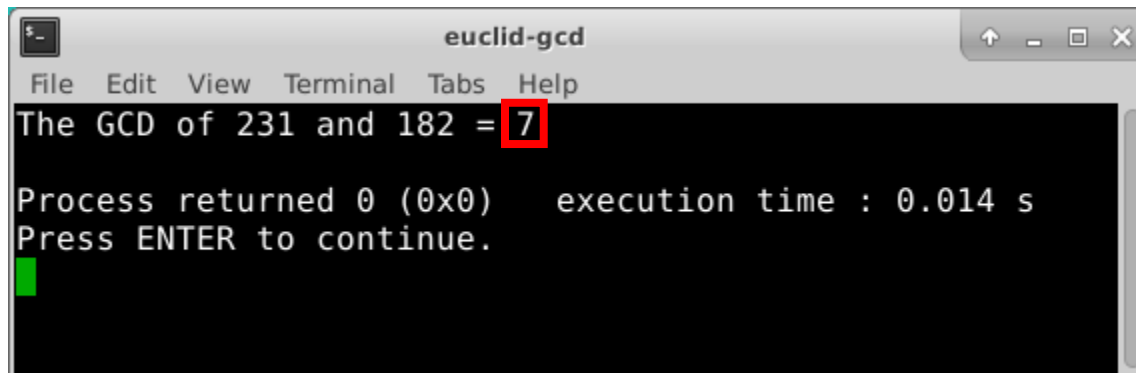**Ensure $a \geq b$**

Edit Lab 2 – Euclid's GCD

```cpp
#include "stdafx.h"

using namespace std;

int GCD(int a, int b)
{
    if (a < b)
        swap(a, b);

    int c = a - b;

    while (c > 0)
    {
        if (c > b)
            a = c;
        else
        {
            a = b;
            b = c;
        }
        c = a -b;
    }

    return b;
}
```

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 ($1 \le n \le 100{,}000$) 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 $\sqrt{\dfrac{6}{probability}}$

Open Lab 3 – Coprime Probability

```cpp
main.cpp

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

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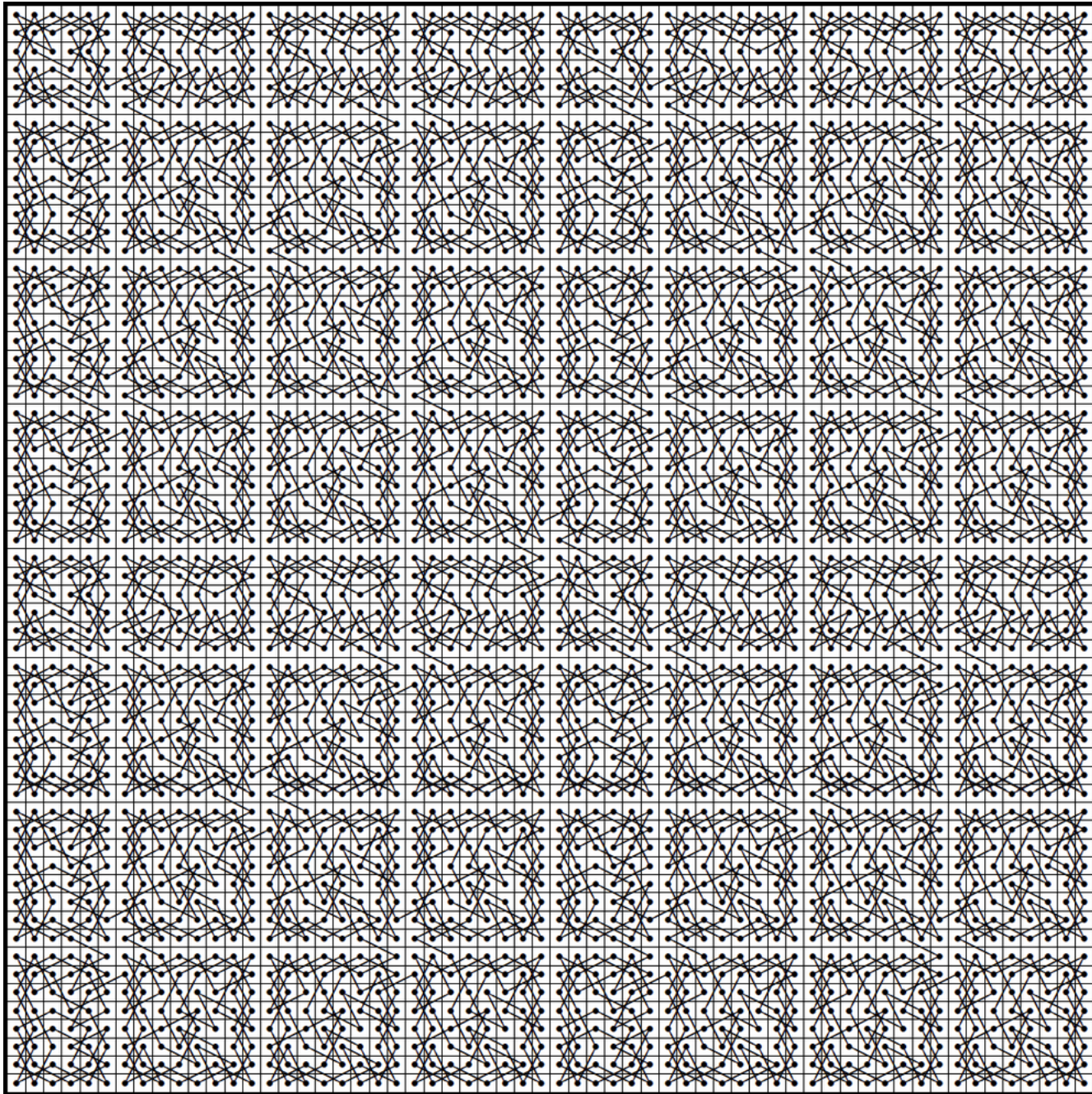
$$0.607927102 \approx 61\% \approx \frac{6}{\pi^2}$$

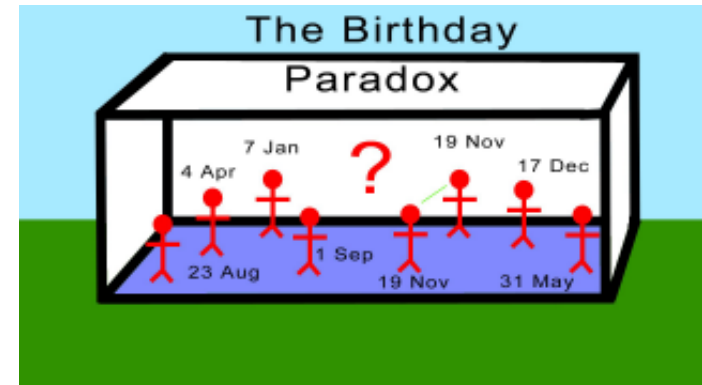$$\frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \cdots = \frac{\pi^2}{6}$$

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?

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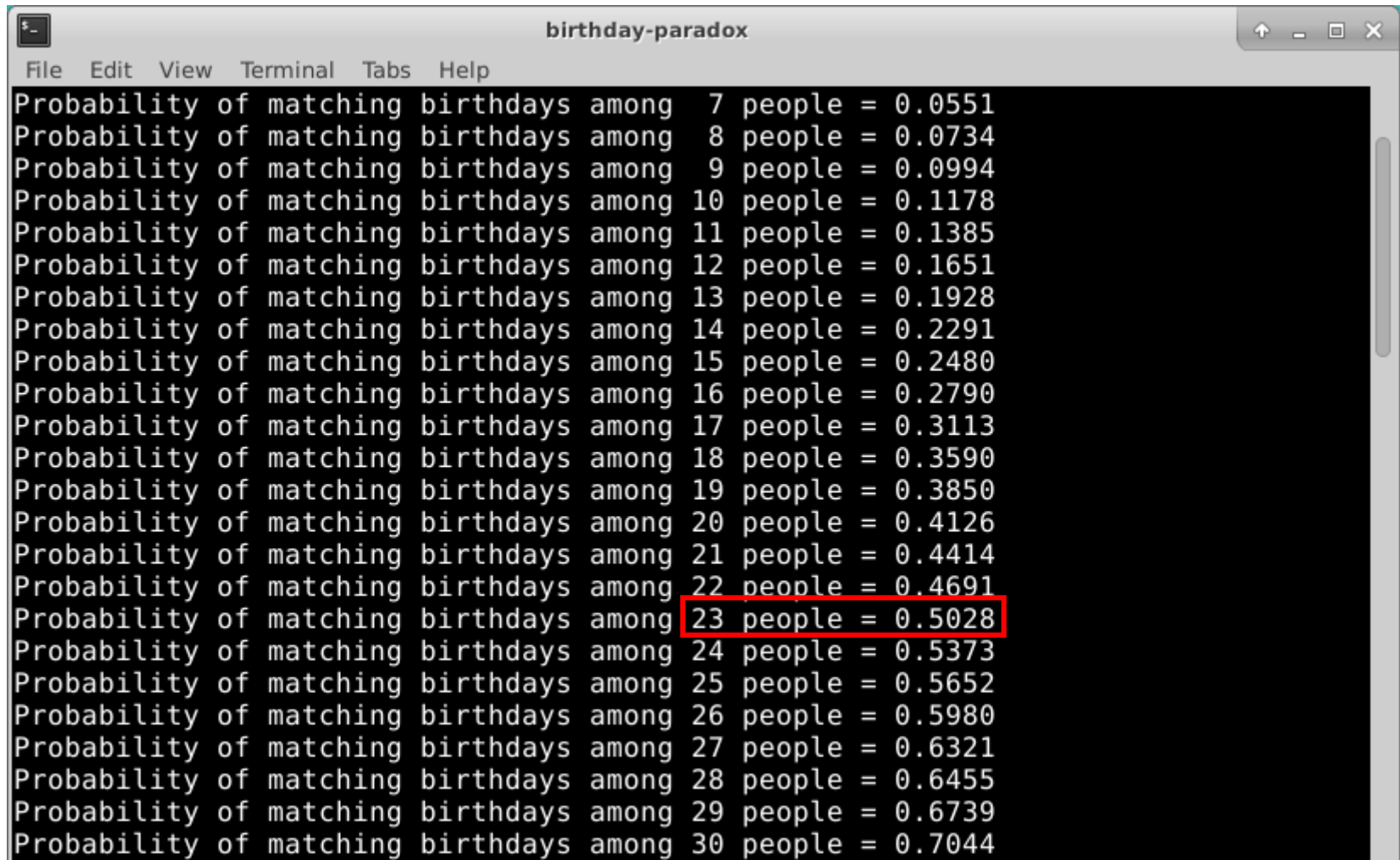
Open
Lab 4
Birthday
Paradox

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

19

```cpp
#include "stdafx.h"

using namespace std;

int main()
{
    seed_seq seed{ 2016 };
    default_random_engine generator{ seed };
    uniform_int_distribution<int> distribution(0, 364);

    for (int students = 2; students <= 80; students++)
    {
        vector<int> birthdays(students);

        int matchCount = 0;

        for (int classNum = 0; classNum < 10000; classNum++)
        {
            // Initialize the birthdays array with a random day between 0 and 364
            for (int i = 0; i < birthdays.size(); i++)
                birthdays.at(i) = distribution(generator);

            // Compare birthdays of each person to the remaining people
            // Note:  Only loop until the first match is found
            bool foundMatch = false;
            for (int j{}; !foundMatch && j < birthdays.size() - 1; j++)
                for (int k{ j + 1 }; !foundMatch && k < birthdays.size(); k++)
                    if (birthdays.at(j) == birthdays.at(k))
                        foundMatch = true;

            if (foundMatch)
                matchCount++;
        }

        cout << "Probability of matching birthdays among "
            << setw(2) << students << " people = "
            << fixed << setprecision(4) << (double)matchCount / 10000
            << endl;
    }

    return 0;
}
```

Edit
Lab 4
Birthday
Paradox

20

Run Lab 4 – Birthday Paradox

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 $= \dfrac{\pi^2}{6}$

- The probability two random integers are coprime $= \dfrac{6}{\pi^2}$

- 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!** ☺