12-2020

Sum of Cubes of the First n Integers

Obiamaka L. Agu

Sum Of Powers Of The First N Integers

A Thesis

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Arts

in

Mathematics

---

by

Obiamaka Agu

December 2020

Sum Of Powers Of The First N Integers

A Thesis

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Obiamaka Agu

December 2020

Approved by:

Hajrudin Fejzić, Ph.D., Committee Chair

Zahid Hasan, Ph.D., Committee Member

Jeffrey Meyer, Ph.D., Committee Member

Madelaine Jetter, Chair, Department of Mathematics

Corey Dunn, Graduate Coordinator, Department of Mathematics

Abstract

In Calculus we learned that $\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$, that $\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$, and that $\sum_{k=1}^{n} k^3 = (\frac{n(n+1)}{2})^2$. These formulas are useful when solving for the area below a quadratic or cubic function over an interval $[a, b]$. This tedious process, solving for areas under a quadratic or a cubic, served as motivation for the introduction of Riemman integrals. For the over zealous math student, these steps were replaced by a simpler method of evaluating antiderivatives at the endpoints a and b. From my recollection, a former instructor informed us do the value of memorizing these formulas. At the very least, we needed to know that the sum of the first n integers is a polynomial of degree 2 with the leading term $\frac{1}{2}n^2$, the sum of the squares of the first n integers is a polynomial of degree 3 with the leading term $\frac{1}{3}n^3$; lastly, the sum of the cubes of the first n integers is a polynomial of degree 4 with the leading term $\frac{1}{4}n^4$. This resulted in the recognition of a pattern; hence, the generalization; the sum of the r-th powers of the first n integers is a polynomial of degree $r+1$ with the leading term $\frac{1}{r+1}n^{r+1}$. Thus, we arrive at the theorem

Theorem 0.0.1. *Let $r \geq 0$ be an integer. Then there is a unique polynomial, $p_r(x)$ of degree $r+1$ with the leading coefficient $\frac{1}{r+1}$ such that for every integer n,*

$$\sum_{k=1}^{n} k^r = p_r(n).$$

To some math scholars, these polynomials are called Faulhaber polynomials, named after Faulhaber, a German mathematician, who was one of the first mathematicians to recognize that the sum of r-th powers is indeed a polynomial. His discoveries resulted in "simple" forms of formulating these polynomials when r is odd.

I will prove Theorem 0.0.1, and many other properties of Faulhaber polynomials. For example, I will prove that for all r the coefficient of x^r is $\frac{1}{2}$. Thus

$$p_r(x) = \frac{1}{r+1}x^{r+1} + \frac{1}{2}x^r + \cdots .$$

The remaining coefficients of this polynomial, do not have a simple form. This is especially the case for the coefficient of x. It turns out that the coefficients of x in $p_r(x)$, are the so called Bernoulli numbers. I will establish this as well as many related properties about Bernoulli numbers.

I will end my thesis with an interesting observation discussed in [Fej05]. The formula for the sum of cubes is especially interesting giving us the identity

$$1^3 + 2^3 + 3^3 + \cdots + (n-2)^3 + (n-1)^3 + n^3 = (1 + 2 + 3 + \cdots + (n-2) + (n-1) + n)^2 .$$

If in both left hand and right-hand sides of this identity we replace the term $(n-1)$ by 2 one would think that the resulting equality

$$1^3 + 2^3 + 3^3 + \cdots + (n-2)^3 + (2)^3 + n^3 = (1 + 2 + 3 + \cdots + (n-2) + (2) + n)^2$$

is false. It was shown in [Fej05] it is actually true. Moreover the pair $((n-2), 2)$ is the only such switch that works for all n.

Acknowledgements

I would, first, like to thank my family for being my biggest cheerleaders and pushing me to pursue an advanced degree. They inspired me to seek out all resources available to advance, not only my career, but myself. Seriously, my father is counting the months, days, hours, and minutes, till graduation. My aunt and uncles in both Iowa, London, and Nigeria are all making plans for my graduation and I have yet to complete the program. My mother, also in Nigeria, constantly inquires, not only about my health, but about school, if I'm getting help enough to complete my courses, and much more. They are intense. I'm very happy and thankful to have been blessed with such loving, caring, and supportive individuals.

The next individuals that I would like to thank are my friends. These individuals are a tough act to follow. They set the bar so high. In moments when I feel inadequate, defeated, and/or ready to give up, they're present with a fresh palm to slap my senses back to reality. They have and continue to provide both physical and emotional support throughout my graduate experience. They are strict, loving, supportive, and uncompromising when it comes to academic and personal growth. I will always be grateful for their continued support.

I would also like to extend a special thank you to my overly enthusiastic husband Buddy. This man is something out of a fairy tale because I would be sweating bullets, with nervousness, for a final and/or presentation and he'll spew some magical words of encouragement and I suddenly feel like I can conquer anything. He was the first to read my work and provide constructive feedback on how I can improve. He's been there to support and encourage me in times when the workload seemed unbearable. I don't know how I would've got to this stage in my academic career without his help, and for that, I am thankful.

Lastly, I would like to give a special thanks to Dr. Hajrudin Fejzić for all his help and support in the development and formation of this paper. The first time that I met him was summer of 2011 in his Combinatorics class. At first I thought this professor was intimidating. What if I don't like his style of teaching? I should just run-out now. However, he was organized, clear and concise, willing to help at any moment, soft spoken, and very knowledgeable. Upon this realization, he easily became one of the professors, at Cal State, that I admired. It was only logical that I'd seek out his assistance upon

my return to school. I, quickly, learned that, although he was an excellent instructor, he proved to be an exemplary advisor. He showed an award winning patience when I made idiotic mathematical errors. He understood that, while working on my thesis, I was completing other courses. He was available during office hours, via email, even though he had other courses to teach. I swear, I don't know how he does it. For all these reasons and more, I will be forever thankful.

There are a multitude of other people, who have contributed to my academic and personal growth, that I would like to express gratitude for. From my high school JROTC instruction Sgt. Edward Brackins, to my former West Point classmates, now officers in the U.S. Army, Shalela Dowdy etc., to my ride or die ex-military bestie Quincy Jallah, and to many other influential individuals that motivated my success. I would like to extend a massive THANK YOU!

Table of Contents

List of Tables

Chapter 1

The Sums

1.1 Carl Friedrich Gauss and the sum of the first n integers

Carl Friedrich Gauss, (1777-1855), born in Brunswick Germany, was a child prodigy considered to be the greatest German mathematician of the 19^{th} century [Wel19]. He was dubbed the "Prince of Mathematics" and his work influenced a great deal of the math areas studied in this day and age which include: number theory, algebra, differential geometry among other fields. At age three, he corrected an error in his father's payroll calculations. By age 7, his arithmetic abilities had grown so much so that his instructors admitted that there was nothing more that they could teach him [Wel19].

In one of his classes, he was given what was meant to be busy work. The problem: find the sum of all the numbers from 1 to 100. He solved this in a matter of minutes. How? Obviously, individually adding the numbers 1 to 100 is incredibly time consuming. If we start by adding the smaller numbers, it'll look something like this:

$$
\begin{aligned}
1+2+3 &= 6\\
1+2+3+4 &= 10\\
1+2+3+4+5 &= 15\\
1+2+3+4+5+6 &= 21\\
1+2+3+4+5+6+7 &= 28\\
1+2+3+4+5+6+7+8 &= 36
\end{aligned}
$$

and so on [Wel19].

We don't know how Gauss calculated the sum of the first 100 numbers; but, he may have recognized that when there are even number of numbers to add as in $1+2+3+4+5+6+7+8$, we could pair 1 with 8, 2 with 7, 3 with 6, and 4 with 5. Each pair adds up to 9, and there are four pairs so that the sum is $9 \cdot 4 = 36$. Similarly, if we want to add the first 100 numbers, we could pair 1 with 100, 2 with 99, 3 with 97, and so on with the last pair being 50 with 51. Each pair sums to 101 and there are 50 pairs. So, the sum of the first 100 numbers is $101 \cdot 50 = 5050$.

Another way to add the first 100 integers, with the little help of symbolic algebra, is as follows. Let S denote the sum. Then,

$$
\begin{array}{rccccccc}
S = & 1 & +2 & +3 & +\cdots+ & 98 & +99 & +100\\
S = & 100 & +99 & +98 & +\cdots+ & 3 & +2 & +1\\
2S = & 101 & +101 & +101 & +\cdots+ & 101 & +101 & +101
\end{array}
$$

The first row is just the definition of S. The right hand side of the second row is the sum of the first 100 integers written backwards, so it is also S. Finally the third row is obtained by adding the corresponding numbers of the two rows above. The left hand side of the third row is $2S$, while the right hand side is $101 \cdot 100$. Hence $2S = 101 \cdot 100$, and from here $S = \frac{101 \cdot 100}{2} = 5050$.

This approach can be easily modified to derive the formula for the sum of the

first n integers. Let S denote the sum. Then,

$$\begin{array}{llllllll} S = 1 & +2 & +3 & +\cdots+ & (n-2) & +(n-1) & +n \\ S = n & +(n-1) & +(n-2) & +\cdots+ & 3 & +2 & +1 \\ 2S = (n+1) & +(n+1) & +(n+1) & +\cdots+ & (n+1) & +(n+1) & +(n+1) \end{array}$$

Similar to that above, the first row is just the definition of S. The right hand side of the second row is the sum of the first n integers written backwards, so it is also S. Finally the third row is obtained by adding the corresponding numbers of the two rows above. The left hand side of the third row is $2S$, while the right hand side is $(n+1) \cdot n$. Hence $2S = (n+1) \cdot n$, and from here $S = \frac{(n+1) \cdot n}{2}$.

This proves the following theorem.

Theorem 1.1.1. *The sum of the first n integers is $\frac{(n+1) \cdot n}{2}$.*

We could generalize this approach to derive the sum of the first n terms of the arithmetic progression $a_0, a_0 + d, a_0 + 2d, \ldots, a_0 + (n-1)d$. Let S denote this sum. Then

$$\begin{array}{llllll} S = & a_0 & +(a_0 + d) & +(a_0 + 2d) & +\cdots & +(a_0 + (n-1)d) \\ S = & (a_0 + (n-1)d) & +(a_0 + (n-2)d) & +(a_0 + (n-3)d) & +\cdots & +a_0 \\ 2S = & (2a_0 + (n-1)d) & +(2a_0 + (n-1)d) & +(2a_0 + (n-1)d) & +\cdots & +(2a_0 + (n-1)d) \end{array}$$

Again, the first row is just the definition of S. The right hand side of the second row is the sum of the first n terms written backwards, so it is also S. Finally the third row is obtained by adding the corresponding numbers of the two rows above. The left hand side of the third row is $2S$, while the right hand side is $(2a_0 + (n-1)d) \cdot n$. Hence $2S = (2a_0 + (n-1)d) \cdot n$, and from here $S = \frac{(2a_0 + (n-1)d) \cdot n}{2}$.

Thus we arrive to the following result.

Theorem 1.1.2. *The sum of the first n terms of the arithmetic progression $a_0, a_0 + d, a_0 + 2d, \ldots$ is $\frac{(2a_0 + (n-1)d) \cdot n}{2}$,*

The special case of Theorem 1.1.2 with $a_0 = 1$ and $d = 2$ is an interesting result in its own.

Corollary 1.1.3. *The sum of the first n odd numbers is n^2.*

1.2 The sum of squares of the first n integers

For the sum of the first n squares, the pattern does not hold. Let

$$\begin{aligned} S_2 &= 1^2 + 2^2 + 3^2 + \cdots + n^2 \\ &= 1 + 4 + 9 + \cdots + n^2. \end{aligned}$$

In order to obtain the formula for S_2, we may be tempted to repeat the same procedure that was successful in the aforementioned computation of:

$$S_1 = 1 + 2 + 3 + \cdots + n.$$

Thus, write the sum twice, reversing the order the second time:

$$\begin{array}{llllllll} S_2 = & 1 & + 4 & + 9 & + \cdots + & (n-2)^2 & +(n-1)^2 & +n^2 \\ S_2 = & n^2 & + (n-1)^2 & + (n-2)^2 & + \cdots + & 9 & +4 & +1 \end{array}$$

Adding both sums results in the sum,

$$2S_2 = [1 + n^2] + [4 + (n-1)^2] + [(n-2)^2 + 9] + \cdots + [(n-1)^2 + 4] + [n^2 + 1]$$

For this method to work in the right-hand side of this equation all the expressions inside the brackets should be identical which obviously is not the case. Thus, we have to consider an alternative approach. First we will redo the formula for the sum, S_1, of the first n integers. We begin by writing

$$\begin{array}{rll} (n+1)^2 = n^2 & + 2n & + 1 \\ n^2 = (n-1)^2 & + 2(n-1) & + 1 \\ (n-1)^2 = (n-2)^2 & + 2(n-2) & + 1 \\ \vdots & \vdots & \vdots \\ 5^2 = 4^2 & + 2 \cdot 4 & + 1 \\ 4^2 = 3^2 & + 2 \cdot 3 & + 1 \\ 3^2 = 2^2 & + 2 \cdot 2 & + 1 \\ 2^2 = 1^2 & + 2 \cdot 1 & + 1 \end{array}$$

After adding these n equations and canceling the numbers $2^2, 3^2, \ldots, n^2$ that show up to the left and to the right of "=" sign, the resulting equation becomes

$$(n+1)^2 = 1^2 + 2(1 + 2 + \cdots + n) + 1 \cdot n$$

or

$$(n+1)^2 = 1 + 2 \cdot S_1 + n.$$

Solving for S_1 we get

$$S_1 = \frac{(n+1)^2 - n - 1}{2}.$$

Mimicking this approach for S_2, the sum of the squares of the first n integers, we use the identity $(a+1)^3 = a^3 + 3a^2 + 3a + 1$ to write

$$\begin{array}{rlll}
(n+1)^3 = n^3 & + 3n^2 & + 3n & + 1 \\
n^3 = (n-1)^3 & + 3(n-1)^2 & + 3(n-1) & + 1 \\
(n-1)^3 = (n-2)^3 & + 3(n-2)^2 & + 3(n-2) & + 1 \\
\vdots & \vdots & \vdots & \vdots \\
4^3 = 3^3 & + 3 \cdot 3^2 & + 3 \cdot 3 & + 1 \\
3^3 = 2^3 & + 3 \cdot 2^2 & + 3 \cdot 2 & + 1 \\
2^3 = 1^3 & + 3 \cdot 1^2 & + 3 \cdot 1 & + 1
\end{array}$$

After adding these n equations and canceling the numbers $2^3, 3^3, \ldots, n^3$ that show up to the left and to the right of "=" sign, the resulting equation becomes

$$(n+1)^3 = 1^3 + 3 \sum_{k=1}^{n} k^2 + 3 \sum_{k=1}^{n} k + 1 \cdot \mathrm{n}$$

or

$$(n+1)^3 = 1 + 3S_2 + 3S_1 + n.$$

Using that $S_1 = \frac{n(n+1)}{2}$ and solving for S_2 results in

$$S_2 = \frac{(n+1)^3 - 1 - 3\frac{n(n+1)}{2} - n}{3}$$

which simplifies to the familiar formula

$$S_2 = \frac{n(n+1)(2n+1)}{6}.$$

Thus we have prove the following theorem

Theorem 1.2.1. *The sum, S_2, of squares of the first n integers is $S_2 = \frac{n(n+1)(2n+1)}{6}$.*

1.3 Sums of cubes and fourth powers of the first n integers

In this section we will find the formulas for S_3, the sum of cubes of the first n integers, and for S_4, the sum of fourth powers of the first n integers. The method is identical with obvious changes to the method used in the previous section to find the formula for S_2.

In order to find the formula for S_3 we begin with the identity $(a+1)^4 = a^4 + 4a^3 + 6a^2 + 4a + 1$ to write the following equations

$$\begin{array}{rlllll}
(n+1)^4 = n^4 & +4n^3 & +6n^2 & +4n & +1 \\
n^4 = (n-1)^4 & +4(n-1)^3 & +6(n-1)^2 & +4(n-1) & +1 \\
(n-1)^4 = (n-2)^4 & +4(n-2)^3 & +6(n-2)^2 & +4(n-2) & +1 \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
4^4 = 3^4 & +4\cdot 3^3 & +6\cdot 3^2 & +4\cdot 3 & +1 \\
3^4 = 2^4 & +4\cdot 2^3 & +6\cdot 2^2 & +4\cdot 2 & +1 \\
2^4 = 1^4 & +4\cdot 1^3 & +6\cdot 1^2 & +4\cdot 1 & +1
\end{array}$$

After adding these n equations and canceling the numbers $2^4, 3^4, \ldots, n^4$ that show up to the left and to the right of "=" sign, the resulting equation becomes

$$(n+1)^4 = 1^4 + 4\sum_{k=1}^{n} k^3 + 6\sum_{k=1}^{n} k^2 + 4\sum_{k=1}^{n} k + 1 \cdot \mathrm{n}$$

or

$$(n+1)^4 = 1 + 4S_3 + 6S_2 + 4S_1 + n.$$

Using that $S_1 = \frac{n(n+1)}{2}$, $S_2 = \frac{n(n+1)(2n+1)}{6}$ and solving for S_3 results in

$$S_3 = \frac{(n+1)^4 - 1 - 6\frac{n(n+1)(2n+1)}{6} - 4\frac{n(n+1)}{2} - n}{4}$$

which can be simplified to

$$S_3 = \frac{n^2(n+1)^2}{4}.$$

Thus we have proved the following theorem

Theorem 1.3.1. *The sum, S_3, of cubes of the first n integers is* $S_3 = \left[\frac{n\cdot(n+1)}{2}\right]^2$.

In order to find the formula for S_4 we begin with the identity $(a+1)^5 = a^5 + 5a^4 + 10a^3 + 10a^2 + 5a + 1$ to write the following equations

$$\begin{array}{rllllll}
(n+1)^5 = n^5 & +5n^4 & +10n^3 & +10n^2 & +5n & +1 \\
n^5 = (n-1)^5 & +5(n-1)^4 & +10(n-1)^3 & +10(n-1)^2 & +5(n-1) & +1 \\
(n-1)^5 = (n-2)^5 & +5(n-2)^4 & +10(n-2)^3 & +10(n-2)^2 & +5(n-2) & +1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
4^5 = 3^5 & +5\cdot 3^4 & +10\cdot 3^3 & +10\cdot 3^2 & +5\cdot 3 & +1 \\
3^5 = 2^5 & +5\cdot 2^4 & +10\cdot 2^3 & +10\cdot 2^2 & +5\cdot 2 & +1 \\
2^5 = 1^5 & +5\cdot 1^4 & +10\cdot 1^3 & +10\cdot 1^2 & +5\cdot 1 & +1
\end{array}$$

After adding these n equations and canceling the numbers $2^5, 3^5, \ldots, n^5$ that show up to the left and to the right of "=" sign, the resulting equation becomes

$$(n+1)^5 = 1^5 + 5\sum_{k=1}^{n} k^4 + 10\sum_{k=1}^{n} k^3 + 10\sum_{k=1}^{n} k^2 + 5\sum_{k=1}^{n} k + 1\cdot \mathrm{n}$$

or

$$(n+1)^5 = 1 + 5S_4 + 10S_3 + 10S_2 + 5S_1 + n.$$

Using that $S_1 = \frac{n(n+1)}{2}$, $S_2 = \frac{n(n+1)(2n+1)}{6}$, and $S_3 = \left[\frac{n(n+1)}{2}\right]^2$ to solve for S_4 results in

$$S_4 = \frac{(n+1)^5 - 10\left[\frac{n(n+1)}{2}\right]^2 - 10\frac{n(n+1)(2n+1)}{6} - 5\frac{n(n+1)}{2} - n - 1}{5}$$

which can be simplified to

$$S_4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}.$$

Thus we have proved the following theorem

Theorem 1.3.2. *The sum, S_4, of the fourth powers of the first n integers is* $S_4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$.

1.4 Sum of the r-th powers of the first n integers is a polynomial

The derivation of formulas for S_2, S_3 and S_4 followed the same pattern, so it is no surprise that these methods can be used to find a formula for S_r, the sum of r-th powers of the first n integers.

Theorem 1.4.1. *Let $r \geq 1$ be an integer. For $1 \leq k \leq r$, let S_k denote the sum of k-th powers of the first n integers. Then*

$$S_r = \frac{(n+1)^{r+1} - 1 - \sum_{k=2}^{r} \binom{r+1}{k} S_{r+1-k} - n}{r+1}.$$

Proof. We begin with the binomial identity $(a+1)^{r+1} = a^{r+1} + \binom{r+1}{1}a^r + \binom{r+1}{2}a^{r-1} + \cdots + \binom{r+1}{r}a + 1$ to write the following equations

$$\begin{array}{rlllll}
(n+1)^{r+1} = n^{r+1} & + \binom{r+1}{1} n^r & + \binom{r+1}{2} n^{r-1} & + \cdots + & \binom{r+1}{r} n & + 1 \\
n^{r+1} = (n-1)^{r+1} & + \binom{r+1}{1} (n-1)^r & + \binom{r+1}{2} (n-1)^{r-1} & + \cdots + & \binom{r+1}{r} (n-1) & + 1 \\
(n-1)^{r+1} = (n-2)^{r+1} & + \binom{r+1}{1} (n-2)^r & + \binom{r+1}{2} (n-2)^{r-1} & + \cdots + & \binom{r+1}{r} (n-2) & + 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
4^{r+1} = 3^{r+1} & + \binom{r+1}{1} 3^r & + \binom{r+1}{2} 3^{r-1} & + \cdots + & \binom{r+1}{r} 3 & + 1 \\
3^{r+1} = 2^{r+1} & + \binom{r+1}{1} 2^r & + \binom{r+1}{2} 2^{r-1} & + \cdots + & \binom{r+1}{r} 2 & + 1 \\
2^{r+1} = 1^{r+1} & + \binom{r+1}{1} 1^r & + \binom{r+1}{2} 1^{r-1} & + \cdots + & \binom{r+1}{r} 1 & + 1
\end{array}$$

After adding these n equations and canceling the numbers $2^{r+1}, 3^{r+1}, \ldots, n^{r+1}$ that show up to the left and to the right of "=" sign, the resulting equation becomes

$$(n+1)^{r+1} = 1 + \binom{r+1}{1} \sum_{k=1}^{n} k^r + \binom{r+1}{2} \sum_{k=1}^{n} k^{r-1} + \cdots + \binom{r+1}{r} \sum_{k=1}^{n} k + n$$

or

$$(n+1)^{r+1} = 1 + (r+1)S_r + \binom{r+1}{2} S_{r-1} + \cdots + \binom{r+1}{r} S_1 + n.$$

Solving the last equation for S_r proves the theorem. □

Corollary 1.4.2. *S_r is a polynomial in n of degree $r+1$ with the leading coefficient $\frac{1}{r+1}$.*

Proof. The proof is by induction on r. For $r = 1$, we have seen that $S_1 = \frac{1}{2}n^2 + \frac{1}{2}n$ so the induction case $r = 1$ is established. Now assume that $r > 1$ and that for all $1 \leq k \leq r-1$, S_k is a polynomial of degree $k+1$. Since by induction hypothesis every term in the right-hand side of the formula for S_r in Theorem 1.4.1 is a polynomial in n, with the term $(n+1)^{r+1}$ being of the highest degree, S_r is also a polynomial of the same degree as $(n+1)^{r+1}$ and its leading coefficient is $\frac{1}{r+1}$. □

Chapter 2

The Numbers

2.1 Bernoulli numbers

In this chapter we will take a different approach. We will not assume any of the results from Chapter 1. Let $r \geq 0$ be an integer and let $P(r)$ denote the set off all polynomials $p(x)$ with $p(0) = 0$ that satisfy the identity

$$p(x+1) - p(x) = (x+1)^r. \tag{2.1}$$

We will show that $P(r)$ has exactly one element which we denote by $p_r(x)$. Moreover, we will show that $p_r(n) = S_r = \sum_{k=1}^{n} k^r$ for all $n \in \mathbb{N}$.

Lemma 2.1.1. *Let $r \geq 0$ be an integer. Suppose that $p(x) \in P(r)$. Then there is a constant A such that*

$$q(x) = (r+1)\int_0^x p(t)dt + Ax \in P(r+1).$$

Proof. Let A be a constant and let

$$q(x) = (r+1)\int_0^x p(t)dt + Ax.$$

Since by assumption $p(x)$ is a polynomial, and since polynomials are closed under integration, $q(x)$ is also a polynomial. Note that $q(0) = 0$. The derivative of q satisfies

$$q'(x) = (r+1)p(x) + A. \tag{2.2}$$

Hence

$$\begin{aligned}(q(x+1)-q(x))' &= q'(x+1)-q'(x) \\ &= (r+1)(p(x+1)-p(x)) && \text{by (2.2)} \\ &= (r+1)(x+1)^r. && \text{by (2.1).}\end{aligned}$$

By taking anti-derivatives we obtain

$$q(x+1)-q(x) = (x+1)^{r+1}+C \tag{2.3}$$

for some constant C. If we take $A = 1-(r+1)\int_0^1 p(t)dt$ then $q(1)=1$. Substituting $x=0$ into (2.3) we obtain

$$\begin{aligned}q(1)-q(0) &= 1+C \\ 1-0 &= 1+C\end{aligned}$$

It follows that $C=0$. Thus for this choice of A, $q(x)\in P(r+1)$. □

Theorem 2.1.2. *For every non-negative integer r, $P(r)$ has exactly one element, $p_r(x)$. Moreover $p_r(n)=\sum_{k=1}^n k^r$ for all $n\in\mathbb{N}$.*

Proof. The proof that $P(r)\neq\emptyset$ is by induction on r. For $r=0$, it is straight forward to check that $x\in P(0)$. Now assume that for some $r\geq 0$, $P(r)\neq\emptyset$. Let $p(x)\in P(r)$. Then by lemma 2.1.1, $P(r+1)\neq\emptyset$ and the induction is complete.

We also use induction this time on n to show that if $p(x)\in P(r)$ then $p(n)=\sum_{k=1}^n k^r$ for all $n\in\mathbb{N}$. For $n=1$ we have to show that $p(1)=1$. Indeed, from (2.1) $p(0+1)-p(0)=(0+1)^r$, and since $p(0)=0$ we get $p(1)=1$. Now assume that for some $n\geq 1$, $p(n)=\sum_{k=1}^n k^r$. From $p(n+1)=p(n)+(n+1)^r=\sum_{k=1}^n k^r+(n+1)^r=\sum_{k=1}^{n+1}k^r$ completing the induction proof.

Now if $t(x)\in P(r)$ then $p(n)-t(n)=0$ for all $n\in\mathbb{N}$ and since $p(x)-t(x)$ is a polynomial this is possible only if $p(x)-t(x)=0$ for all $x\in\mathbb{R}$. Hence $P(r)$ has only one element. □

From lemma 2.1.1 we have that $p_{r+1}(x)=(r+1)\int_0^x p(t)dt+Ax$ for some constant A. From (2.2) we see that $A=p'_{r+1}(0)$. These constants, $\{A_r=p'_r(0)\}_{r=0}^{\infty}$ are known in literature as Bernoulli numbers. We will show that Bernoulli numbers, $\{A_r\}$, can be generated independent of polynomials, $\{p_r(x)\}$, and that coefficients of these polynomials can be expressed in terms of Bernoulli numbers.

Lemma 2.1.3. *For $k \geq 2$, the k-th derivative of $p_{r+1}^{(k)}(x) = (r+1)p_r^{(k-1)}(x)$.*

Proof. This is immediate from (2.2). □

In the statement of next lemma for $k > r+1$, the subscript of A_{r+1-k} is negative, and Bernoulli numbers are not defined for negative subscripts. We could restrict the value of k to $k \leq r+1$ to avoid this, or we could simply set $A_t = 0$ for negative t, and that is what we are going to do.

Lemma 2.1.4. *Let $k \geq 1$ be an integer. The k-th derivative of $p_r(x)$ at 0 satisfies the following formula*

$$\frac{p_r^{(k)}(0)}{k!} = \frac{1}{r+1}\binom{r+1}{k}A_{r+1-k}.$$

In particular $\frac{p_r^{(k)}(0)}{k!} = 0$ for $k > r+1$.

Proof. For $k = 1$ this formula just means that $p_r'(0) = \frac{1}{r+1}\binom{r+1}{1}A_{r+1-1} = A_r$ which is true by definition, so it needs only to be verified for all $k \geq 2$. The proof is by induction on r. For $r = 0$, $p_0(x) = x$, so that all its derivatives but the first are zero. We also have that for $k \geq 2$ $A_{0+1-k} = 0$ by definition and the formula is verified for $r = 0$.

Now assume that for some $r \geq 0$ and for all $k \geq 1$, $\frac{p_r^{(k)}(0)}{k!} = \frac{1}{r+1}\binom{r+1}{k}A_{r+1-k}$. Let $k \geq 2$. By lemma 2.1.3

$$\begin{aligned}\frac{p_{r+1}^{(k)}(0)}{k!} &= \frac{r+1}{k}\frac{p_r^{(k-1)}(0)}{(k-1)!} \text{ and by induction hypothesis we get}\\ &= \frac{r+1}{k}\frac{1}{r+1}\binom{r+1}{k-1}A_{r+1-(k-1)}\\ &= \frac{1}{k}\binom{r+1}{k-1}A_{r+2-k} = \frac{1}{r+2}\binom{r+2}{k}A_{r+2-k}\end{aligned}$$

and the induction is complete. That $\frac{p_r^{(k)}(0)}{k!} = 0$ for $k > r+1$, follows from the formula and the definition of A_t for t negative. □

In the proof of Theorem 2.1.5 below we will use the fact that $p_0(x) = x$, and $p_1(x) = \frac{1}{2}x^2 + \frac{1}{2}x$ so that $A_0 = 1$ and $A_1 = \frac{1}{2}$.

Theorem 2.1.5. *The polynomials $p_r(x)$ can be expressed in terms of Bernoulli numbers as $p_r(x) = \frac{1}{r+1}\sum_{k=1}^{r+1}\binom{r+1}{k}A_{r+1-k}\ x^k$. In addition the two leading terms of $p_r(x)$ are $\frac{1}{r+1}x^{r+1}$ and $\frac{1}{2}x^r$.*

Proof. Since $p_r(0) = 0$, by Taylor formula

$$p_r(x) = \sum_{k=1}^{\text{degree of } p_r} \frac{p_r^{(k)}(0)}{k!} x^k.$$

By lemma 2.1.4, $\frac{p_r^{(k)}(0)}{k!} = 0$ for $k > r+1$, and $\frac{p_r^{(r+1)}(0)}{(r+1)!} = \frac{1}{r+1} A_0 = \frac{1}{r+1}$ so that degree of p_r is $r+1$. Also by Lemma 2.1.4, $\frac{p_r^{(k)}(0)}{k!} = \frac{1}{r+1}\binom{r+1}{k} A_{r+1-k}$, proving the first part of Theorem.

For the second part of the Theorem it just remains to show that $\frac{p_r^{(r)}(0)}{r!} = \frac{1}{2}$. To that end by Lemma 2.1.4 with $k = r$ we get
$\frac{p_r^{(r)}(0)}{r!} = \frac{1}{r+1}\binom{r+1}{r} A_{r+1-r} = A_1 = \frac{1}{2}$. □

Theorem 2.1.5 and the fact that $p_r(1) = 1$ for all r, allows us to express Bernoulli numbers recursively independent of $p_r(x)$.

Corollary 2.1.6. *Bernoulli numbers can be defined recursively as* $A_0 = 1$ *and* $A_r = 1 - \frac{1}{r+1}\sum_{n=0}^{r-1}\binom{r+1}{n} A_n$ *for* $r \geq 1$.

Proof. From $p_r(1) = 1$ and Theorem 2.1.5 we have

$$1 = p_r(1) = \frac{1}{r+1}\sum_{k=1}^{r+1}\binom{r+1}{k} A_{r+1-k}$$

using substitution $n = r + 1 - k$, and that $\binom{r+1}{k} = \binom{r+1}{r+1-n} = \binom{r+1}{n}$, we can rewrite this as

$$\begin{aligned} 1 &= \frac{1}{r+1}\sum_{n=0}^{r}\binom{r+1}{n} A_n \text{ or} \\ &= \frac{1}{r+1}\sum_{n=0}^{r-1}\binom{r+1}{n} A_n + A_r. \end{aligned}$$

Solving for A_r finishes the proof. □

Here is the list of the first twenty one Bernoulli numbers obtained using Mathematica code:

Input:

A[0]=1

Table[A[r] = 1 - 1/(r + 1)*Sum[Binomial[r + 1, n]*A[n], {n, 0, r - 1}], {r, 0, 20}]

Output:

1, 1/2, 1/6, 0, -(1/30), 0, 1/42, 0, -(1/30), 0, 5/66, 0, -(691/ 2730), 0, 7/6, 0, -(3617/510), 0, 43867/798, 0, -(174611/330)

Here is the list of $p_r(x)$ for $r = 0, 1, 2, \ldots, 20$.

$$
\begin{aligned}
p_0(x) &= x \\
p_1(x) &= \frac{1}{2}x^2 + \frac{1}{2}x \\
p_2(x) &= \frac{1}{3}x^3 + \frac{1}{2}x^2 + \frac{1}{6}x \\
p_3(x) &= \frac{1}{4}x^4 + \frac{1}{2}x^3 + \frac{1}{4}x^2 \\
p_4(x) &= \frac{1}{5}x^5 + \frac{1}{2}x^4 + \frac{1}{3}x^3 - \frac{1}{30}x \\
p_5(x) &= \frac{1}{6}x^6 + \frac{1}{2}x^5 + \frac{5}{12}x^4 - \frac{1}{12}x^2 \\
p_6(x) &= \frac{1}{7}x^7 + \frac{1}{2}x^6 + \frac{1}{2}x^5 - \frac{1}{6}x^3 + \frac{1}{42}x \\
p_7(x) &= \frac{1}{8}x^8 + \frac{1}{2}x^7 + \frac{7}{12}x^6 - \frac{7}{24}x^4 + \frac{1}{12}x^2 \\
p_8(x) &= \frac{1}{9}x^9 + \frac{1}{2}x^8 + \frac{2}{3}x^7 - \frac{7}{15}x^5 + \frac{2}{9}x^3 - \frac{1}{30}x \\
p_9(x) &= \frac{1}{10}x^{10} + \frac{1}{2}x^9 + \frac{3}{4}x^8 - \frac{7}{10}x^6 + \frac{1}{2}x^4 - \frac{3}{20}x^2 \\
p_{10}(x) &= \frac{1}{11}x^{11} + \frac{1}{2}x^{10} + \frac{5}{6}x^9 - x^7 + x^5 - \frac{1}{2}x^3 + \frac{5}{66}x \\
p_{11}(x) &= \frac{1}{12}x^{12} + \frac{1}{2}x^{11} + \frac{11}{12}x^{10} - \frac{11}{8}x^8 + \frac{11}{6}x^6 - \frac{11}{8}x^4 + \frac{5}{12}x^2 \\
p_{12}(x) &= \frac{1}{13}x^{13} + \frac{2}{13}x^{12} - \frac{29}{78}x^{11} - \frac{11}{6}x^9 + \frac{22}{7}x^7 - \frac{33}{10}x^5 + \frac{5}{3}x^3 - \frac{691}{2730}x \\
p_{13}(x) &= \frac{1}{14}x^{14} + \frac{1}{84}x^{12} + \frac{13}{30}x^{10} + \frac{143}{28}x^8 - \frac{143}{20}x^6 + \frac{65}{12}x^4 - \frac{691}{420}x^2 \\
p_{14}(x) &= \frac{1}{15}x^{15} + \frac{1}{30}x^{14} + \frac{11}{150}x^{11} + \frac{143}{18}x^9 - \frac{143}{10}x^7 + \frac{91}{6}x^5 - \frac{691}{90}x^3 + \frac{7}{6}x \\
p_{15}(x) &= \frac{1}{16}x^{16} - \frac{277}{672}x^{10} - \frac{429}{16}x^8 + \frac{455}{12}x^6 - \frac{691}{24}x^4 + \frac{35}{4}x^2 \\
p_{16}(x) &= \frac{1}{17}x^{17} + \frac{1}{357}x^{11} - \frac{319}{255}x^9 + \frac{260}{3}x^7 - \frac{1382}{15}x^5 + \frac{140}{3}x^3 - \frac{3617}{510}x \\
p_{17}(x) &= \frac{1}{18}x^{18} + \frac{1}{12}x^{17} + \frac{1}{252}x^{12} + \frac{17}{270}x^{10} + \frac{1105}{6}x^8 - \frac{11747}{45}x^6 + \frac{595}{3}x^4 - \frac{3617}{60}x^2 \\
p_{18}(x) &= \frac{1}{19}x^{19} + \frac{7}{114}x^{17} - \frac{8}{95}x^{11} - \frac{5770}{627}x^9 - \frac{23494}{35}x^7 + 714x^5 - \frac{3617}{10}x^3 + \frac{43867}{798}x \\
p_{19}(x) &= \frac{1}{20}x^{20} - \frac{19}{40}x^{19} - \frac{1}{120}x^{18} + \frac{39}{88}x^{10} - \frac{13437877}{54600}x^8 + 2261x^6 - \frac{68723}{40}x^4 + \frac{43867}{84}x^2 \\
p_{20}(x) &= \frac{1}{21}x^{21} + \frac{1}{9}x^{19} + \frac{5}{154}x^{11} + \frac{678562}{28665}x^9 + 6460x^7 - \frac{68723}{10}x^5 + \frac{219335}{63}x^3 - \frac{174611}{330}x
\end{aligned}
$$

Notice that for $r > 1$ and r odd, all A_r's in the list are zero. That this is always true is a part of Theorem 2.1.7 below.

Theorem 2.1.7. *For r odd, we have $p_r(x) - p_r(-x) = x^r$. It follows that $A_r = 0$ for $r > 1$ and odd.*

Proof. First we will show that for r odd, $p_r(x) - p_r(-x) = x^r$ for all $x \in \mathbb{R}$. Since $p_r(x) - p_r(-x)$ is a polynomial it is enough to verify this for $x = n$ for all $n \in \mathbb{N}$. We begin by substituting $x = n-1$, $x = n-2, \cdots, x = -n$ into (2.1) to obtain the following identities:

$$\begin{aligned}
p_r(n) &= p_r(n-1) + n^r \\
p_r(n-1) &= p_r(n-2) + (n-1)^r \\
p_r(n-2) &= p_r(n-3) + (n-2)^r \\
&\vdots \\
p_r(2) &= p_r(1) + 2^r \\
p_r(1) &= p_r(0) + 1^r \\
p_r(0) &= p_r(-1) + 0^r \\
p_r(-1) &= p_r(-2) - 1^r \\
p_r(-2) &= p_r(-3) - 2^r \\
&\vdots \\
p_r(-(n-2))) &= p_r(-(n-1)) - (n-2)^r \\
p_r(-(n-1)) &= p_r(-n) - (n-1)^r.
\end{aligned}$$

In the bottom half of these identities we used that r is odd so that $(-k)^r = -k^r$. By adding these identities, and taking into account that all the terms except $p_r(n)$, n^r and $p_r(-n)$ cancel, we obtain $p_r(n) = p_r(-n) + n^r$ or $p_r(n) - p_r(-n) = n^r$.

Next we will show that $A_r = 0$ for $r > 1$ and odd. To that end we first differentiate $p_r(x) - p_r(-x) = x^r$, to get $p_r'(x) + p_r'(-x) = rx^{r-1}$. Substitution $x = 0$ into the last equation yield $A_r = p_r'(0) = 0$ for $r > 1$ and the proof is complete. □

Corollary 2.1.8. *If $r > 1$ is odd, then $p_{r-1}(x) = \frac{p_r'(x)}{r}$.*

Proof. Let $r > 1$ and r odd. Let $p(x) = \frac{p_r'(x)}{r}$. In order to show that $p_{r-1}(x) = p(x)$ we need to show that $p(x) \in P(r-1)$. That $p(x)$ is a polynomial is obvious. By Theorem 2.1.7, $A_r = 0$ so that $p(0) = 0$. It remains to show that $p(x+1) - p(x) = (x+1)^{r-1}$, but this follows immediately by differentiating $p_r(x+1) - p_r(x) = (x+1)^r$ and dividing the resulting equation through by r. □

Corollary 2.1.9. *If $r \neq 0$ even, then $p_r(x) + p_r(-x) = x^r$.*

Proof. By Theorem 2.1.7 $p_{r+1}(x) - p_{r+1}(-x) = x^{r+1}$. Now differentiating the last equation we get $p_{r+1}'(x) + p_{r+1}'(-x) = (r+1)x^r$. Dividing through by $r+1$ we obtain $\frac{p_{r+1}'(x)}{r+1} + \frac{p_{r+1}'(-x)}{r+1} = x^r$. But by corollary 2.1.7 $p_r(x) = \frac{p_{r+1}'(x)}{r+1}$ proving that $p_r(x) + p_r(-x) = x^r$. □

We already know that $p_r(x)$ is a polynomial of degree $r+1$ and that the coefficients of x^{r+1} and x^r are $\frac{1}{r+1}$ and $\frac{1}{2}$ respectively. The next corollary shows that "every other" coefficient is zero.

Corollary 2.1.10. *For r odd, $p_r(x) - \frac{1}{r+1}x^{r+1} - \frac{1}{2}x^r$ is of the form $\sum_{j=0} a_j x^{r-1-2j}$, while for r even, $p_r(x) - \frac{1}{r+1}x^{r+1} - \frac{1}{2}x^r$ is of the form $\sum_{j=0} a_j x^{r-2-2j}$.*

Proof. For r odd we have to prove that $p^{(k)}(0) = 0$ for $k \neq r$ and k odd. This follows by differentiating k-times the identity $p_r(x) - p_r(-x) = x^r$ from Theorem 2.1.7 to obtain $p_r^{(k)}(0) + p_r(k)(0) = 0$.

For r even we have to prove that $p^{(k)}(0) = 0$ for $k \neq r$ and k even. This follows by differentiating k-times the identity $p_r(x) + p_r(-x) = x^r$ from corollary 2.1.7 to obtain $p_r^{(k)}(0) + p_r^{(k)}(0) = 0$. □

2.2 Faulhaber

Bernoulli's formula is sometimes called Faulhaber's formula. Johann Faulhaber of Ulm was a notable German mathematician whose research was particularly significant in providing alternative methods of computing the sums of powers. In his findings he recognized the patterns in the sums which led to the formulation of polynomial identities that express the sum powers of the first n integers. The resulting polynomials have been extensively studied by a multitude of math scholars. However, Faulhaber's main contribution to the study of these polynomials was that he was the first one to realize that for r odd, the coefficients of $p_r(x)$ can be expressed in terms of $x(x+1)$. We will end this chapter by proving this. In what follows we are not assuming any results that we proved so far.

First we need the following lemma regarding odd polynomials. A polynomial, $p(x)$ is odd if $p(-x) = p(x)$. Since odd polynomials are closed under addition and scalar multiplication, the set of all odd polynomials, O, is a subspace of the set of all polynomials. The natural basis for the vector space of odd polynomials is $\{x^{2j-1}\}_{j=1}^{\infty}$. It is also true that the set of all odd polynomials of degree less than or equal to $2k-1$ is a subspace of O with the dimension equal to k.

Lemma 2.2.1. *Let r be an odd integer. The sequence $\{x^j\left((x+1)^j-(x-1)^j\right)\}_{j=1}^{\frac{r+1}{2}}$ of odd polynomials forms the basis for all odd polynomials of degree less than or equal to r.*

Proof. Let $q_j(x) = x^j\left((x+1)^j-(x-1)^j\right)$. Then

$$\begin{aligned} q_j(-x) &= (-x)^j\left((-x+1)^j-(-x-1)^j\right) \\ &= (-1)^j x^j\left((-1)^j(x-1)^j-(-1)^j(x+1)^j\right) \\ &= x^j\left((x-1)^j-(x+1)^j\right) = -q_j(x) \end{aligned}$$

proving that $q_j(x)$ are odd polynomials. The degree of $q_j(x)$ is $2j-1$, so that the set $W = \{q_j\}_{j=1}^{\frac{r+1}{2}}$ is linearly independent, and since W has $\frac{r+1}{2}$ elements it is the basis for the space of all odd polynomials of degree less than or equal to r. □

If r is an odd integer then x^r is an odd polynomial so by Lemma 2.2.1, x^r is a linear combination of $\{x^j\left((x+1)^j-(x-1)^j\right)\}_{j=1}^{\frac{r+1}{2}}$.

Theorem 2.2.2. *Let $r = 2m-1$. If $x^r = \sum_{k=1}^{m} a_k x^k\left((x+1)^k-(x-1)^k\right)$, then $p_r(x) = \sum_{k=1}^{m} a_k x^k (x+1)^k$.*

Proof. Let $p(x) = \sum_{k=1}^{m} a_k x^k (x+1)^k$. Then $p(-x) = \sum_{k=1}^{m} a_k (-1)^k x^k (-1)^k (x-1)^k$ so that $p(x) - p(-x) =$
$\sum_{k=1}^{m} a_k x^k \left((x+1)^k - (x-1)^k\right) = x^r$. In particular

$$p(x+1) - p(-(x+1)) = (x+1)^r.$$

But $p(-(x+1)) = \sum_{k=1}^{m} a_k (-(x+1))^k ((-(x+1)+1)^k$
$= \sum_{k=1}^{m} a_k (-1)^k (x+1)^k (-1)^k (x)^k = p(x)$. Hence $p(x+1) - p(x) = (x+1)^r$. Since $p(0) = 0$, $p(x) \in P(r)$, completing the proof.

□

Thus for r odd, Faulhaber polynomial can indeed be expressed as a polynomial in $x(x+1)$. For r even by Corollary 2.1.8 we have the following result.

Corollary 2.2.3. *Let r be even. If $p_{r+1}(x) = \sum_{k=1}^{\frac{r+2}{2}} a_k x^k (x+1)^k$, then $p_r(x) = \frac{2x+1}{r+1} \sum_{j=0}^{\frac{r}{2}} a_{j+1}(j+1)x^j(x+1)^j$.*

Proof. By corollary 2.1.8, $p_r(x) = \frac{p'_{r+1}(x)}{r+1}$. But $p'_{r+1}(x) = \frac{d}{dx} \sum_{k=1}^{\frac{r+2}{2}} a_k (x^2+x)^k$, so by Chain rule, $p'_{r+1}(x) = (2x+1) \sum_{k=1}^{\frac{r+2}{2}} a_k k (x^2+x)^{k-1}$. Let $j = k-1$, then $p'_{r+1}(x) = (2x+1) \sum_{j=0}^{\frac{r}{2}} a_{j+1}(j+1)(x^2+x)^j$. Dividing through by $r+1$ completes the proof. □

Here is the list of $p_r(x)$ expressed in the terms of $x(x+1)$ for the first 10 odd values of r. The list is generated using C++ codes illustrated in Appendix A.

$$
\begin{aligned}
p_1(x) &= \frac{1}{2}x(x+1) \\
p_3(x) &= \frac{1}{4}(x(x+1))^2 \\
p_5(x) &= \frac{1}{6}(x(x+1))^3 - \frac{1}{12}(x(x+1))^2 \\
p_7(x) &= \frac{1}{8}(x(x+1))^4 - \frac{1}{6}(x(x+1))^3 + \frac{1}{12}(x(x+1))^2 \\
p_9(x) &= \frac{1}{10}(x(x+1))^5 - \frac{1}{4}(x(x+1))^4 + \frac{3}{10}(x(x+1))^3 - \frac{3}{20}(x(x+1))^2 \\
p_{11}(x) &= \frac{1}{12}(x(x+1))^6 - \frac{1}{3}(x(x+1))^5 + \frac{17}{24}(x(x+1))^4 - \frac{5}{6}(x(x+1))^3 \\
&\quad + \frac{5}{12}x(x+1))^2 \\
p_{13}(x) &= \frac{1}{14}(x(x+1))^7 - \frac{5}{12}(x(x+1))^6 + \frac{41}{30}(x(x+1))^5 - \frac{59}{21}(x(x+1))^4 \\
&\quad + \frac{691}{210}(x(x+1))^3 + \frac{691}{420}(x(x+1))^2 \\
p_{15}(x) &= \frac{1}{16}(x(x+1))^8 - \frac{1}{2}(x(x+1))^7 + \frac{7}{3}(x(x+1))^6 - \frac{22}{3}(x(x+1))^5 \\
&\quad + \frac{359}{24}(x(x+1))^4 - \frac{35}{2}(x(x+1))^3 + \frac{35}{4}(x(x+1))^2 \\
p_{17}(x) &= \frac{1}{18}(x(x+1))^9 - \frac{7}{12}(x(x+1))^8 + \frac{11}{3}(x(x+1))^7 - \frac{293}{18}(x(x+1))^6 \\
&\quad + \frac{1519}{30}(x(x+1))^5 - \frac{1237}{12}(x(x+1))^4 + \frac{3617}{30}(x(x+1))^3 - \frac{3617}{60}(x(x+1))^2 \\
p_{19}(x) &= \frac{1}{20}(x(x+1))^{10} - \frac{2}{3}(x(x+1))^9 + \frac{217}{40}(x(x+1))^8 - \frac{1129}{35}(x(x+1))^7 \\
&\quad + \frac{2829}{20}(x(x+1))^6 - \frac{6583}{15}(x(x+1))^5 + \frac{750167}{840}(x(x+1))^4 - \frac{43867}{42}(x(x+1))^3 \\
&\quad + \frac{43867}{84}(x(x+1))^2
\end{aligned}
$$

Chapter 3

The Discovery

3.1 On sum of cubes

We have found formulas, S_r, for the sums of r-th powers of the first n integers for the first twenty values of r. An interesting and well-known observation is that $S_3 = S_1^2$ that is

$$1^3 + 2^3 + \cdots + n^3 = (1 + 2 + \cdots + n)^2. \tag{3.1}$$

What may come as a surprise is that if for $n \geq 2$, in (3.1), we replace $(n-1)$ by 2 on the both sides of the equations the resulting equation

$$1^3 + 2^3 + \cdots + (n-2)^3 + 2^3 + n^3 = (1 + 2 + \cdots + (n-2) + 2 + n)^2 \tag{3.2}$$

is still true. This remarkable identity has been studied in [Fej05]. In this paper the authors consider a more general question of finding all triples (k, x, n) with $1 \leq k, x < n$ for which if k in (3.1) is replaced by $x \neq k$ the resulting identity

$$\sum_{j=1}^{k-1} j^3 + x^3 + \sum_{j=k+1}^{n} j^3 = (\sum_{j=1}^{k-1} j + x + \sum_{j=k+1}^{n} j)^2 \tag{3.3}$$

still holds. The main result in [Fej05] is the following theorem.

Theorem 3.1.1. *Let k, x, and n be positive integers such that $1 \leq k, x \leq n$, $n \neq k$. Then the triple (k, x, n) satisfies (3.3) if and only if*
(a) $x = k$ or $(k, x, n) = ((n-1), 2, n)$ or
(b) $(k, x, n) = (2(p-1) + \frac{3(p-1)p}{s}, 2p + s, x + k - p)$ where integers p and s satisfy $p \geq 2$

and s is a positive divisor of $3p(p-1)$. Moreover, different values of the pair (p, s) yield different solutions.

We can actually use Theorem 3.1.1 to not only find a triple (k, x, n) satisfying equation (3.3) but also to determine if a given triple is in fact, a solution.
For example: if $p = 20$, then we calculate $3(p-1)p = 1140$, and for $s = 10$ we obtain the triple $(152, 50, 182)$ while if we pick another positive divisor of 1140, say $s = 19$, we obtain the triple (98, 59, 137).
Hence,

$$\sum_{j=1}^{151} j^3 + 50^3 + \sum_{j=153}^{182} j^3 = 273,935,601 = (\sum_{j=1}^{151} j + 50 + \sum_{j=153}^{182} j)^2$$

and

$$\sum_{j=1}^{97} j^3 + 59^3 + \sum_{j=99}^{137} j^3 = 88,623,396 = (\sum_{j=1}^{97} j + 59 + \sum_{j=99}^{137} j)^2.$$

On the other hand, (97, 37, 130) is not a solution because this triple would yield $p = x + k - n = 98 + 37 - 130 = 5$ and thus, $s = x - 2p = 37 - 10 = 27$ would have to be a divisor of $3(p-1)p = 60$.
When we check the validity of these values we get,

$$\sum_{j=1}^{97} j^3 + 37^3 + \sum_{j=99}^{130} j^3 = 71,614,686$$
$$\neq 71,470,116 = (\sum_{j=1}^{97} j + 37 + \sum_{j=99}^{130} j)^2.$$

Theorem 3.1.1 part b) naturally raises questions about what else can be said about the "non-trivial" triples (k, x, n). Obviously from Theorem 3.1.1 part b) the formulas for k and x give us the lower bounds for k and x. Namely $k \geq 2(p-1)+1 \geq 2(2-1)+1 = 3$ and $x = 2p + s \geq 2p + 1 \geq 5$.

What if $k \geq 3$ is given, can we find x and n so that (k, x, n) is a "non-trivial" solution to (3.3). Or if $x \geq 5$ is given, can we find k and n so that (k, x, n) is a "non-trivial" solution to (3.3). The answer to both questions is yes.

First if $k \geq 3$ is given, and odd we can take $p = \frac{k+1}{2}$ and $s = 3(p-1)p$ so that $k = 2(p-1) + \frac{3(p-1)p}{s}$. If $k \geq 3$ is given and even we can take $p = \frac{k}{2}$ and $s = 3\frac{(p-1)p}{2}$ so that $k = 2(p-1) + \frac{3(p-1)p}{s}$.

In the case $x \geq 5$ is given and odd we can take $p = \frac{x-1}{2}$ and $s = 1$ so that $x = 2p + s$. If $x \geq 5$ is given and even we can take $p = \frac{x}{2} - 1$ and $s = 2$ so that $x = 2p + s$.

An obvious question is what if n is given. Can we find k and x so that (k, x, n) is a "non-trivial" solution to (3.3). In this case the answer is yes if and only if $n^2 + n + 1$ is not a prime number or 3 times a prime number.[fej05]

We end this chapter by listing all "non-trivial" solution to (3.3) for all $2 \leq p \leq 153$. The following table was generated with the help of C++ code detailed in Appendix B.

Table 3.1: Nontrivial Solution Chart

	k	x	n		k	x	n		k	x	n
p = 2	4	7	9	p=8	26	30	48	p=13	33	78	98
	5	6	9		22	37	51		28	143	158
	3	10	11		21	40	53		26	260	273
	8	5	11		20	44	56		25	494	506
p=3	6	15	18		18	58	68	p=14	47	54	87
	13	8	18		17	72	81		40	67	93
	7	12	16		16	100	108		39	70	95
	10	9	16		15	184	191		33	106	125
	6	15	18	p=9	24	45	60		32	119	137
	5	24	26		20	72	83		29	210	225
p=4	12	14	9		18	126	135		28	301	315
	6	15	18		17	234	242		27	574	587
	10	17	23	p=10	28	47	65	p=15	46	65	96
	8	26	30		23	74	87		42	75	102
	7	44	47		20	155	165		38	93	116
p=5	12	20	29		19	290	299		37	100	122
	14	20	29	p=11	35	44	68		35	120	140
	18	16	29		31	52	72		33	156	174
	13	22	30		30	55	74		30	345	360
	12	25	32		26	77	92		29	660	674
	11	30	36		25	88	102	p=16	50	68	102
	10	40	45		23	132	144		48	72	104
	9	70	74		22	187	198		46	77	107
p=6	10	22	35		21	352	362		40	104	128
	20	21	35	p=12	40	46	74		39	112	135
	15	30	39		33	60	81		38	122	144
	12	57	63		31	68	87		35	176	195
	11	102	107		26	123	137		34	212	230
p=7	21	28	42		24	222	234		32	392	408
	19	32	44		23	420	431		31	752	767
	14	77	84	p=13	42	52	81	p=17	56	68	107
	13	140	146		37	62	86		49	82	114

Table 3.2: Nontrivial Solution Chart

	k	x	n		k	x	n		k	x	n
p=17	48	85	116	p=22	64	107	149	p=26	60	247	281
	44	102	129		60	121	159		56	377	407
	40	136	159		56	143	177		55	442	471
	38	170	191		53	170	201		53	702	729
	36	238	257		51	198	227	p=27	78	135	186
	35	306	324		49	242	269		65	216	254
	34	442	459		44	737	759	p=28	82	137	191
	33	850	866	p=23	77	92	146		68	218	258
p=18	61	70	113		67	112	156		61	380	413
	51	90	123		66	115	158		58	623	653
	36	495	513		55	184	216	p=29	98	116	185
	35	954	971		50	299	326		85	142	198
p=19	63	76	120		47	552	576		84	145	200
	55	92	128		46	805	828		77	174	222
	38	551	570	p=24	82	94	152		70	232	273
p=20	68	78	126		69	120	165		68	261	300
	58	97	135		64	140	180		63	406	440
	57	100	137		55	232	263		62	464	497
	53	116	149		54	255	285		60	667	698
	50	135	165		50	462	488		59	870	900
	48	154	182		48	876	900	p=30	103	118	191
	44	230	254	p=25	88	95	158		87	150	207
	43	268	291		84	100	159		76	205	251
	42	325	347		73	122	170		68	321	359
	41	420	441		68	140	183		67	350	387
	40	610	630		66	150	191		63	582	615
p=21	76	77	132		58	230	263	p=31	105	124	198
	75	78	132		57	250	282		91	152	212
	68	87	134		56	275	306		78	217	264
	60	105	144		53	410	438		70	341	380
	58	112	149		52	500	527		69	372	410
	54	132	165		50	950	975		65	620	654
	50	168	197	p=26	89	102	165	p=32	110	126	204
	49	182	210		80	117	171		94	157	219
	47	222	248		76	127	177		93	160	221
	45	294	318		75	130	179		86	188	242
	44	357	380		65	182	221		78	250	296
	42	672	693		63	202	239		74	312	354

Table 3.3: Nontrivial Solution Chart

	k	x	n		k	x	n		k	x	n
p=32	70	436	474	p=37	126	148	237	p=42	91	658	707
	68	560	596		109	182	254		89	822	869
	66	808	842		99	222	284	p=43	147	172	276
p=33	108	138	213	p=38	131	150	243		127	212	296
	100	154	221		112	187	261		102	387	446
	96	165	228		111	190	263		98	473	528
	86	210	263		93	298	353		93	688	738
	82	242	291		80	779	821		91	860	908
	80	264	311	p=39	114	195	270	p=44	152	174	282
	75	354	396		102	249	312		130	217	303
	73	418	458		95	312	368		129	220	305
	72	462	501		94	325	380		119	260	335
	68	858	893		89	420	470		108	346	410
p=34	100	167	233		85	572	618		98	561	615
	88	221	275	p=40	143	152	255		97	604	657
	84	255	305		130	170	260	p=45	143	198	296
	83	266	315		123	184	267		142	200	297
	77	374	417		118	197	275		132	225	312
	75	442	483		114	210	284		115	310	380
p=35	119	140	224		104	260	324		110	360	425
	110	155	230		98	314	372		108	387	450
	103	172	240		96	340	396		99	630	684
	102	175	242		91	440	491		98	684	737
	98	189	252		88	548	596	p=46	144	207	305
	89	240	294		87	600	647		136	227	317
	85	280	330		86	665	711		117	322	393
	83	308	356	p=41	140	164	263		113	362	429
	82	325	372		121	202	282		100	713	767
	78	427	470		120	205	284	p=47	161	188	302
	75	580	620		110	246	315		139	232	324
	74	665	704		104	287	350		138	235	326
	73	828	867		100	328	387		115	376	444
p=36	124	142	230		95	410	464	p=48	166	190	308
	105	180	249		92	492	543		141	240	333
	98	207	269		90	574	623		130	284	366
	97	212	273		88	697	744		112	472	536
	90	261	315		86	902	947		110	519	581
	84	342	390	p=42	145	166	269		103	848	903
	80	450	494		123	210	291		102	942	996
	77	612	653		100	371	429	p=49	168	196	315
	75	828	867		96	453	507		159	210	320

Table 3.4: Nontrivial Solution Chart

	k	x	n		k	x	n		k	x	n
p=49	152	224	327	p=54	187	214	347	p=59	203	236	380
	145	242	338		159	270	375		175	292	408
	132	294	377	p=55	189	220	354		174	295	410
	124	350	425		163	272	380		145	472	558
	114	490	555		130	515	590	p=60	208	238	386
	112	539	602		119	920	984		177	300	417
	110	602	663	p=56	198	217	359		163	356	459
	105	882	938		194	222	360		154	415	509
p=50	173	198	321		187	232	363		138	651	729
	168	205	323		180	244	368		136	710	786
	148	247	345		176	252	372	p=61	210	244	393
	147	250	347		170	266	380		181	302	422
	140	275	365		166	277	387		165	366	470
	133	310	393		165	280	389		156	427	522
	128	345	423		154	322	420		140	671	750
	123	394	467		152	332	428		138	732	809
	119	450	519		150	343	437	p=62	215	246	399
	113	590	653		145	376	465		184	307	429
	112	625	687		143	392	479		183	310	431
	108	835	893		140	420	504		153	490	581
p=51	185	192	326		138	442	524	p=63	186	315	438
	150	255	354		134	497	575		178	343	458
	145	272	366		132	532	608		155	504	596
	134	327	410		131	552	627		151	560	648
	125	408	482		130	574	648	p=64	234	240	410
	118	527	594		125	728	797		190	317	443
	117	552	618		124	772	840		182	344	462
	110	867	926		122	882	948		180	352	468
p=52	170	221	339	p=57	188	240	371		158	506	600
	154	257	359		184	247	374		154	560	650
	138	325	411		175	266	384		153	576	665
	136	338	422		168	285	396		142	884	962
	128	410	486		150	366	459	p=65	232	250	417
	120	546	614		148	380	471		224	260	419
	119	572	639		140	456	539		208	286	429
	115	716	779		131	618	692		206	290	431
p=53	182	212	341		130	646	719		193	322	450
	157	262	366		126	798	867		192	325	452
	156	265	368	p=58	172	287	401		188	338	461
	143	318	408		152	377	471		180	370	485
	130	424	501		143	458	543		176	390	501

Table 3.5: Nontrivial Solution Chart

	k	x	n		k	x	n		k	x	n
p=65	160	520	615	p=71	245	284	458	p=77	229	382	534
	158	546	639		211	352	492		228	385	536
	154	610	699		210	355	494		218	420	561
	152	650	737		182	497	608		209	462	594
	148	754	837		175	568	672		196	553	672
	144	910	989		170	639	738		194	532	689
p=66	240	249	423		161	852	942		190	616	729
	229	262	425	p=72	250	286	464		185	686	794
	220	275	429		213	360	501		180	781	884
	195	330	459		196	428	552	p=78	280	299	501
	185	366	485		169	712	809		271	310	503
	175	418	527	p=73	252	292	471		253	338	513
	156	627	717		217	362	506		245	354	521
	152	717	803		198	438	563		231	390	543
	148	847	929		171	730	828		217	442	581
p=67	231	268	432	p=74	257	294	477		180	849	951
	199	332	464		220	367	513	p=79	273	316	510
	154	737	824		219	370	515		235	392	548
	150	871	954		183	586	695		182	869	972
p=68	236	270	438	p=75	238	335	498	p=80	278	318	516
	202	337	471		222	375	522		238	397	555
	201	340	473		198	483	606		237	400	557
	185	404	521		193	520	638		218	476	614
	168	538	638		185	600	710		206	555	681
p=69	228	291	450		173	816	914		198	634	752
	204	345	480	p=76	250	323	497		188	792	900
	182	444	557		245	332	501	p=81	240	405	564
	172	529	632		240	342	506		200	648	767
	170	552	653		226	377	527	p=82	244	407	569
	159	750	840		200	494	618		203	650	771
p=70	253	266	449		195	532	651	p=83	287	332	536
	228	301	459		188	602	714		247	412	576
	208	347	485		186	627	737		246	415	578
	201	370	501		175	836	935		205	664	786
	184	455	569	p=77	285	286	494	p=84	292	334	542
	183	462	575		284	287	494		249	420	585
	173	554	657		266	308	497		229	500	645
	161	770	861		236	363	522		202	749	867

Table 3.6: Nontrivial Solution Chart

	k	x	n		k	x	n		k	x	n
p=85	308	323	546	p=91	315	364	588	p=96	262	572	738
	294	340	549		310	371	590		235	800	939
	287	350	552		271	452	632	p=97	336	388	627
	258	408	581		250	533	692		289	482	674
	253	422	590		245	560	714		264	582	749
	238	476	629		234	637	780	p=98	341	390	633
	236	485	636	p=92	338	345	591		292	487	681
	231	510	656		320	366	594		291	490	683
	213	646	774		274	457	639		243	778	923
	204	765	884		273	460	641	p=99	350	387	638
	203	782	900		266	483	657		294	495	690
	202	800	917		260	506	674		273	576	750
p=86	299	342	555		251	548	707		250	737	888
	272	387	573		234	667	809		245	792	938
	256	427	597		228	730	866	p=100	333	420	653
	255	430	599		224	782	914		308	470	678
	221	602	737		221	828	957		306	475	681
	213	682	809	p=93	308	393	608		298	497	695
	204	817	935		276	465	648		253	740	893
p=87	258	435	606		246	600	753		252	750	902
	230	561	704		230	744	881		248	794	942
	215	696	824	p=94	280	467	653	p=101	350	404	653
p=88	290	374	576		248	611	765		301	502	702
	273	408	593		233	746	885		300	505	704
	262	437	611	p=95	329	380	614		275	606	780
	246	495	653		302	425	632		260	707	866
	232	572	716		283	472	660		250	808	957
	218	698	828		282	475	662	p=102	355	406	659
	210	814	936		245	660	810		303	510	711
p=89	308	356	575		235	760	900	p=103	357	412	666
	265	442	618	p=96	350	363	617		307	512	716
	264	445	620		342	372	618	p=104	362	414	672
	242	534	687		334	382	620		310	517	723
	220	712	843		285	480	669		309	520	725
p=90	313	358	581		280	496	680		284	620	800
	267	450	627		270	534	708		258	826	980
	232	625	767		266	552	722	p=105	388	392	675

Table 3.7: Nontrivial Solution Chart

	k	x	n		k	x	n		k	x	n
p=105	348	444	687	p=113	392	452	731	p=122	363	610	851
	338	462	695		337	562	786	p=123	366	615	858
	334	470	699		336	565	788		326	795	998
	325	490	710		308	678	873	p=124	410	527	813
	312	525	732	p=114	397	454	737		370	617	863
	299	570	764		339	570	795	p=125	434	500	809
	298	574	767	p=115	418	437	740		403	550	828
	280	665	840		399	460	744		398	560	833
	278	678	851		343	572	800		373	622	870
	273	714	882		323	644	852		372	625	872
	271	730	896		318	667	870		348	715	938
	264	795	954	p=116	404	462	750		341	750	966
	260	840	995		375	508	767	p=126	439	502	815
p=106	336	477	707		368	522	774		425	522	821
	316	527	737		346	577	807		385	602	861
	300	583	777		345	580	809		375	630	879
	280	689	863		322	667	873	p=127	441	508	822
	273	742	909		317	692	893		379	632	884
	263	842	999		299	812	995	p=128	446	510	828
p=107	371	428	692	p=117	348	585	816		382	637	891
	319	532	744		340	611	834		381	640	893
	318	535	746	p=118	352	587	821		350	764	986
p=108	376	430	698	p=119	413	476	770	p=129	428	546	845
	321	540	753		355	592	828		400	602	873
	295	644	831		354	595	830		384	645	900
p=109	378	436	705		338	651	870	p=130	473	494	837
	325	542	758	p=120	418	478	776		388	647	905
	297	654	842		408	492	780		375	690	935
p=110	383	438	711		391	520	791	p=131	455	524	848
	328	547	765		378	546	804		391	652	912
	327	550	767		374	555	809		390	655	914
	284	765	939		364	580	824	p=132	460	526	854
p=111	405	420	714		357	600	837		393	660	921
	330	555	774		328	716	924	p=133	473	518	858
	319	592	800		323	744	947		462	532	861
	310	629	828	p=121	438	462	779		435	574	876
	294	717	900		420	484	783		418	608	893
p=112	370	476	734		361	602	842		397	662	926
	366	483	737		350	638	867		390	684	941
	348	520	756		339	682	900	p=134	467	534	867
	334	557	779		330	726	935		400	667	933

Table 3.8: Nontrivial Solution Chart

	k	x	n
p=135	402	675	942
p=136	440	596	900
	432	612	908
	406	677	947
p=137	476	548	887
	409	682	954
	408	685	956
p=138	481	550	893
	411	690	963
p=139	483	556	900
	415	692	968
p=140	488	558	906
	418	697	975
	417	700	977
p=141	515	534	908
	468	597	924
	460	611	930
	420	705	984
p=142	424	707	989
p=143	497	572	926
	427	712	996
	426	715	998
p=144	502	574	932
	494	585	935
	462	639	957
p=145	520	560	935
	504	580	939
p=146	509	582	945
	464	657	975
p=148	546	555	953
	490	629	971
p=149	518	596	965
p=150	523	598	971
p=151	525	604	978
p=152	530	606	984
p=153	520	629	996

Chapter 4

Conclusion

I was able to locate a minimum of one solution for all p values up to 153 with the restriction that p≥2 depicted in Theorem 3.1.1. It's also important to note that x≠k, for non-trivial solutions, 1≤k, and x≤n. All triples above satisfy identity 3.3.

We can also observe that, from the resulting values, the sum of the k and x values will always be larger than the corresponding n value. There will always be a solution when x≥5 and k≥3. Also, based on the above charts, we can observe that, for some p values, there exist multiple triple solutions. It is important to note that there are solutions in which k = 2p. Moreover, the p-value is never listed as either a k, x, or n value and all the solutions for k, x, and n are always larger than the p-value.

Lastly, for the n value, there were no recognizable patterns that would aid in predicting future solutions aside from the noticeable fact that the values are increasing as the p-value increases even though there are some repetitions in values of n. However, previous researchers discovered that we can obtain multiple non-trivial solutions by putting some bounds on n. For instance, an online journal published by The Mathematical Association of America on the sum of cubes provided the $n^2 + n + 1 \neq q$ and $n^2 + n + 1 \neq 3q$ for some arbitrary q then we have a non-trivial solution. These equations were instrumental in generating the solution charts above. However, there were other solutions, obtained, that did not always align with the above parameters for q and $3q$; thus, we cannot concretely stand on a definite pattern for n.

Appendix A

How to Generate Faulhaber's Triangle

How To Generate Faulhaber's Triangle

The C++ code provided generates Faulhaber's Triangle values up to row 34 when you see the phrase for (int i = 0; i < 34; i++) in the actual code. The entire code was created by a website, Rosetta Code. The link is provided in the bibliography page. An alternative code is also provided in the Java language. Any experienced coder can copy and paste the code into the program and the triangle will generate. Feel free to extend the triangle beyond row 34.

Language C++
Translation of: C(Hashtag)
Uses C++ 17

```
#include <exception>
#include <iomanip>
#include <iostream>
#include <numeric>
#include <sstream>
#include <vector>

class Frac {
public:
Frac(long n, long d) {
if (d == 0) {
throw new std::runtime_error("d must not be zero");
}

long nn = n;
long dd = d;
if (nn == 0) {
dd = 1;
} else if (dd < 0) {
nn = -nn;
dd = -dd;
}

long g = abs(std::gcd(nn, dd));
if (g > 1) {
nn /= g;
dd /= g;
}
```

```
num = nn;
denom = dd;
}

Frac operator-() const {
return Frac(-num, denom);
}

Frac operator+(const Frac& rhs) const {
return Frac(num*rhs.denom + denom * rhs.num, rhs.denom*denom);
}

Frac operator-(const Frac& rhs) const {
return Frac(num*rhs.denom - denom * rhs.num, rhs.denom*denom);
}

Frac operator*(const Frac& rhs) const {
return Frac(num*rhs.num, denom*rhs.denom);
}

friend std::ostream& operator<<(std::ostream&, const Frac&);

static Frac ZERO() {
return Frac(0, 1);
}

private:
long num;
long denom;
};
```

```
std::ostream & operator<<(std::ostream & os, const Frac &f) {
if (f.num == 0 || f.denom == 1) {
return os << f.num;
}

std::stringstream ss;
ss << f.num << "/" << f.denom;
return os << ss.str();
}

Frac bernoulli(int n) {
if (n < 0) {
throw new std::runtime_error("n may not be negative or zero");
}

std::vector<Frac> a;
for (int m = 0; m <= n; m++) {
a.push_back(Frac(1, m + 1));
for (int j = m; j >= 1; j--) {
a[j - 1] = (a[j - 1] - a[j]) * Frac(j, 1);
}
}

// returns 'first' Bernoulli number
if (n != 1) return a[0];
return -a[0];
}

int binomial(int n, int k) {
if (n < 0 || k < 0 || n < k) {
throw new std::runtime_error("parameters are invalid");
}
```

```
if (n == 0 || k == 0) return 1;

int num = 1;
for (int i = k + 1; i <= n; i++) {
num *= i;
}

int denom = 1;
for (int i = 2; i <= n - k; i++) {
denom *= i;
}

return num / denom;
}

std::vector<Frac> faulhaberTraingle(int p) {
std::vector<Frac> coeffs;

for (int i = 0; i < p + 1; i++) {
coeffs.push_back(Frac::ZERO());
}

Frac q{ 1, p + 1 };
int sign = -1;
for (int j = 0; j <= p; j++) {
sign *= -1;
coeffs[p - j] = q * Frac(sign, 1) * Frac(binomial(p + 1, j), 1) * bernoulli(j);
}

return coeffs;
}
```

```
int main() {
using namespace std;

for (int i = 0; i < 34; i++) {
vector<Frac> coeffs = faulhaberTraingle(i);
for (auto it = coeffs.begin(); it != coeffs.end(); it++) {
cout << right << setw(5) << *it << "  ";
}
cout << endl;
}

return 0;
}
```

Output

$$
\begin{array}{ccccccccccccccc}
1 \\
\frac{1}{2} & & \frac{1}{2} \\
\frac{1}{6} & & \frac{1}{2} & & \frac{1}{3} \\
0 & & \frac{1}{4} & & \frac{1}{2} & & \frac{1}{4} \\
-\frac{1}{30} & & 0 & & \frac{1}{3} & & \frac{1}{2} & & \frac{1}{5} \\
0 & & -\frac{1}{12} & & 0\frac{5}{12} & & \frac{1}{2} & & \frac{1}{6} \\
\frac{1}{42} & & 0 & & -\frac{1}{6} & & 0 & & \frac{1}{2} & & \frac{1}{2} & & \frac{1}{7} \\
0 & & \frac{1}{12} & & 0 & & -\frac{7}{24} & & 0 & & \frac{7}{12} & & \frac{1}{2} & & \frac{1}{8} \\
-\frac{1}{30} & & 0 & & \frac{2}{9} & & 0 & & -\frac{7}{15} & & 0 & & \frac{2}{3} & & \frac{1}{2} & & \frac{1}{9} \\
0 & & -\frac{3}{20} & & 0 & & \frac{1}{2} & & 0 & & -\frac{7}{10} & & 0 & & \frac{3}{4} & & \frac{1}{2} & & \frac{1}{10} \\
\frac{5}{66} & & 0 & & -\frac{1}{2} & & 0 & & 1 & & 0 & & -1 & & 0 & & \frac{5}{6} & & \frac{1}{2} & & \frac{1}{11} \\
0 & & \frac{5}{12} & & 0 & & -\frac{11}{8} & & 0 & & \frac{11}{6} & & 0 & & -\frac{11}{8} & & 0 & & \frac{11}{12} & & \frac{1}{2} & & \frac{1}{12} \\
-\frac{691}{2730} & & 0 & & \frac{5}{3} & & 0 & & -\frac{33}{10} & & 0 & & \frac{22}{7} & & 0 & & -\frac{11}{6} & & 0 & & -\frac{29}{78} & & \frac{2}{13} & & \frac{1}{13} \\
0 & & -\frac{691}{420} & & 0 & & \frac{65}{12} & & 0 & & -\frac{143}{20} & & 0 & & \frac{143}{28} & & 0 & & \frac{13}{30} & & 0 & & \frac{1}{84} & & 0 \\
\frac{1}{14} \\
\frac{7}{6} & & 0 & & -\frac{691}{90} & & 0 & & \frac{91}{6} & & 0 & & -\frac{143}{10} & & 0 & & \frac{143}{18} & & 0 & & \frac{11}{150} & & 0 & & 0 \\
\frac{1}{30} & & \frac{1}{15} \\
0 & & \frac{35}{4} & & 0 & & -\frac{691}{24} & & 0 & & \frac{455}{12} & & 0 & & -\frac{429}{16} & & 0 & & -\frac{277}{672} & & 0 & & 0 & & 0 \\
0 & & 0 & & \frac{1}{16} \\
-\frac{3617}{510} & & 0 & & \frac{140}{3} & & 0 & & -\frac{1382}{15} & & 0 & & \frac{260}{3} & & 0 & & -\frac{319}{255} & & 0 & & \frac{1}{357} & & 0 & & 0 & & 0 \\
0 & & 0 & & \frac{1}{17}
\end{array}
$$

Output Continued

0	$-\frac{3617}{60}$	0	$\frac{595}{3}$	0	$-\frac{11747}{45}$	0	$\frac{1105}{6}$	0
$\frac{17}{270}$	0	$\frac{1}{252}$	0	0	0	0	$\frac{1}{12}$	$\frac{1}{18}$
$\frac{43867}{798}$	0	$-\frac{3617}{10}$	0	714	0	$-\frac{23494}{35}$	0	
$-\frac{5770}{627}$	0	$-\frac{8}{95}$	0	0	0	0	0	
$\frac{7}{114}$	0	$\frac{1}{19}$						
0	$\frac{43867}{84}$	0	$-\frac{68723}{40}$	0	2261	0		
$-\frac{13437877}{54600}$	0	$\frac{39}{88}$	0	0	0	0	0	0
0	$-\frac{1}{120}$	$-\frac{19}{40}$	$\frac{1}{20}$					
$-\frac{174611}{330}$	0	$\frac{219335}{63}$	0	$-\frac{68723}{10}$	0			
6460	0	$\frac{678562}{28665}$	0	$\frac{5}{154}$	0	0	0	
0	0	0	0	$\frac{1}{9}$	0	$\frac{1}{21}$		
0	$-\frac{1222277}{220}$	0	$\frac{219335}{12}$	0	$-\frac{481061}{20}$			
0	$\frac{1421}{132}$	0	$-\frac{71173}{15015}$	0	0	0	$-\frac{1}{660}$	0
0	0	0	0	0	0	$\frac{1}{22}$		
$\frac{854513}{138}$	0	$-\frac{1222277}{30}$	0	$\frac{482537}{6}$	0			
$-\frac{755953}{10}$	0	$\frac{3647}{138}$	0	$-\frac{691}{62790}$	0	0	0	0
0	$\frac{2}{483}$	0	$-\frac{1}{690}$	0	$\frac{1}{138}$	$-\frac{1}{46}$	$\frac{1}{23}$	
0	$\frac{854513}{12}$	0	$-\frac{28112371}{120}$	0	$\frac{11098351}{36}$	0	$\frac{2166583}{720}$	
0	$\frac{469}{144}$	0	$-\frac{691}{21840}$	0	0	0	0	0
0	0	0	0	$-\frac{1}{48}$	0	$\frac{1}{24}$		

Output Continued

$$
\begin{array}{cccccccc}
-\frac{236364091}{2730} & 0 & \frac{1709026}{3} & 0 & -\frac{28112371}{25} & 0 & -\frac{1163966978}{1425} & 0 \\
\frac{2781473}{2125} & 0 & \frac{21}{10} & 0 & \frac{691}{68250} & 0 & 0 & 0 \\
\frac{1}{250} & 0 & 0 & 0 & 0 & 0 & \frac{1}{150} & \\
-\frac{1}{25} & \frac{1}{25} & & & & & & \\
0 & -\frac{1181820455}{1092} & 0 & \frac{21362825}{6} & 0 & -\frac{28112371}{6} & 0 & \\
-\frac{119976245}{1596} & 0 & \frac{97659}{2210} & 0 & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -\frac{1}{260} & 0 & -\frac{1}{156} & 0 & \frac{1}{26} & \\
\frac{8553103}{6} & 0 & -\frac{1181820455}{126} & 0 & \frac{55543345}{3} & 0 & -\frac{2086776061}{2970} & 0 \\
-\frac{6272981}{21546} & 0 & -\frac{112127}{13770} & 0 & 0 & 0 & 0 & 0 \\
-\frac{5}{594} & 0 & \frac{1}{810} & 0 & -\frac{1}{1134} & 0 & 0 & 0 \\
0 & 0 & & & & & & \\
\frac{1}{27} & & & & & & & \\
0 & \frac{76977927}{4} & 0 & -\frac{3545461365}{56} & 0 & \frac{166630035}{2} & 0 & \\
-\frac{551945371}{1540} & 0 & -\frac{17590667}{22344} & 0 & -\frac{3617}{7140} & 0 & 0 & 0 \\
0 & 0 & 0 & & & & & \\
0 & 0 & 0 & -\frac{1}{1176} & 0 & 0 & & \\
0 & 0 & 0 & \frac{1}{28} & & & & \\
-\frac{23749461029}{870} & 0 & 179615163 & 0 & -\frac{709092273}{2} & 0 & -\frac{61342070218}{2001} & 0 \\
-\frac{15365768}{435} & 0 & \frac{131601}{3857} & 0 & 0 & 0 & 0 & 0 \\
\frac{691}{39585} & 0 & \frac{5}{319} & 0 & 0 & 0 & -\frac{1}{1218} & 0 \\
0 & 0 & \frac{1}{174} & 0 & \frac{1}{29} & & & \\
\end{array}
$$

Output Continued

$$
\begin{array}{cccccc}
0 & -\frac{23749461029}{60} & 0 & \frac{5208839727}{4} & 0 & -\frac{6854558639}{4} \\
-\frac{988671541}{828} & 0 & \frac{523833}{275} & 0 & 0 & 0 \\
0 & \frac{691}{81900} & 0 & 0 & 0 & 0 \\
-\frac{1}{1260} & 0 & 0 & 0 & \frac{1}{180} & -\frac{1}{60} \\
\frac{8615841276005}{14322} & 0 & -\frac{23749461029}{6} & 0 & \frac{15626519181}{2} & 0 \\
\frac{533216112}{713} & 0 & -\frac{1396888}{5115} & 0 & 0 & 0 \\
0 & -\frac{7601}{84630} & 0 & \frac{5}{2046} & 0 & \frac{1}{930} \\
0 & \frac{8615841276005}{924} & 0 & -\frac{736233291899}{24} & 0 & \frac{161474031537}{4} \\
\frac{82963795941}{1120} & 0 & \frac{123904385}{4416} & 0 & 0 & 0 \\
0 & 0 & \frac{5}{2112} & 0 & \frac{1}{960} & 0 \\
\frac{739383942691048263}{408707654270131982} & 0 & \frac{68926730208040}{693} & 0 & -\frac{2944933167596}{15} & 0 \\
0 & \frac{673874023441}{45045} & 0 & \frac{543781}{414} & 0 & 0 \\
0 & -\frac{3617}{8415} & 0 & -\frac{14}{33} & 0 & 0 \\
0 & 0 & 0 & 0 & -\frac{1}{990} & 0 \\
-\frac{1152921504606846976}{1403877295228465437} & \frac{5952926035095041063}{817415308540263964} & 0 & \frac{17231682552010}{21} & 0 & -\frac{16197132421778}{15} \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{5}{1122} & 0 & 0 & 0 & -\frac{1}{1428} \\
0 & 0 & 0 & \frac{1}{204} & 0 & \frac{1}{34}
\end{array}
$$

Appendix B

How To Generate Non Trivial Solutions

How To Generate NonTrivial Solutions

These codes were generated using the aforementioned parameters given for $p \geq 2$, $1 \leq k$, $x \leq n$, $x \neq k$. These rules are those implemented for the resulting identity 3.3. This section provides the C++ codes needed to be able to generate non-trivial solutions. A C++ knowledgeable person is well aware of files that end in .h and those that end in .cpp.

All the codes for each individual file is displayed with their titles. Be mindful that all .h and .cpp files must be open in order for the solutions to generate in main.cpp. I took the liberty of detailing the codes for each independent file on separate pages to avoid any confusion. Once you've copied and pasted the codes into C++ and made minor modifications when you see this phrase (move up a line) or (move up to the previous line). To avoid cutting off the codes at the end of each line I moved it to the next line. Make sure to move it back up before running the program. Once the modifications have been made, run the program. You should a long list of solutions as previously displayed above.

You can extend the solutions further than what I provided by modifying the value listed but be mindful that there will be a considerable wait before an output is received.

Individual.cpp Code For File:

```
//Individual.cpp
#include "Individual.h"

Individual::Individual()
: SumOfCubes() {}

inline void Individual::produceCombinations(int arr[], int data[], const int &start,
const int &end, const int &index, const int &combinations){} (move up a line)

//Function to set k value
//Good!
inline void Individual::kSetter(int arr[]){
    cout << "Please enter a value for k: ";
    cin >> this->k;
    cout << endl;
    xSetter(arr);
}



//Function to set x value
//Good!
inline void Individual::xSetter(int arr[]){
    cout << "Please enter a value for x: ";
    cin >> this->x;
    cout << endl;
    nSetter(arr);
}

//Function to set n value
//Good!
```

```
inline void Individual::nSetter(int arr[]){
    cout << "Please enter a value for n: ";
    cin >> this->n;
    cout << endl;
    while(calc_NBoundaries() == false){
        cout << "n does not work! Please enter a new n: ";
        cin >> this->n;
        cout << endl;
    }
    if(Boundaries() == true){
        checkEquality();
    }
    else{
        kSetter(arr);
    }
}

//Function to test if n is possible
inline bool Individual::calc_NBoundaries(){
    int count(0);
    int q = (GetN_value()*GetN_value() + GetN_value() + 1);
    if(q == 0){
        cout << "q cannot be equal to 0:\n";
        return false;
    }
    if(q == 3){
        cout << "q cannot be equal to 3:\n";
        return false;
    }
    if(q % 3 == 0){
        cout << "q cannot be divisible by 3:\n";
        return false;
```

```
    }
    // for(unsigned int i = 2; i < q; ++i){
    //     if(q % i == 0){
    //         count++;
    //     }
    // }
    // if(count > 1){
    //     cout << "q is not prime:\n";
    //     return false;
    // }
    else{
        return true;
    }
}

//Function to test if numbers are correct
bool Individual::Boundaries(){

    long double p = (GetX_value() + GetK_value() - GetN_value());
        if(GetK_value() != GetN_value()){
            //Removed stipulation that x <= n for testing purposes
            if(GetX_value() != GetN_value()){ //&& GetX_value() <= GetN_value()){
                long double xPlus_K = GetX_value() + GetK_value();
                if(xPlus_K > GetN_value()){
                    if(GetK_value() >= 3){
                        if(GetX_value() >= 5){
                            long double nMinusOne = GetN_value() - 1;
                            if(GetK_value() != nMinusOne){
                                if( p >= 2){
                                    cout << "These values work!\n";
                                    return true;
                                }
```

```
                    else{
                        cout << " p, which is the sum of (k,x,n) is not
                        greater than or equal to 2.\n"; (move up a line)
                        return false;
                    }

                }
                else{
                    cout << "k value is equal to n value minus 1.\n";
                    return false;
                }
            }
            else{
                cout << "x value is not greater than or equal to 5.\n";
                return false;
            }
        }
        else{
            cout << "k value is not greater than or equal to 3.\n";
            return false;
        }
    }
    else{
        cout << "x + K is less than n value.\n";
        return false;
    }
}
else{
    if(GetX_value() == GetN_value()){
        cout << "x value is equal to n value.\n";
        return false;
    }
```

```
                else{
                    cout << "x value is not less than or equal to n value.\n";
                    return false;
                }
            }
        }
        else{
            cout << "k value is equal to n value.\n";
            return false;
        }

    return false;
}

inline bool Individual::checkEquality(){

    long double LHS = (Summation_KLS() + X_valueLS() + Summation_NLS());
    if(LHS == Right_HSquare()){
        cout << "The combination of this (k,x,n) - (";
        cout << GetK_value() <<", " << GetX_value() << ", " << GetN_value() << ")
        works!\n"; (move up a line)
        return true;
    }
    else{
    cout << "The combination of this (k,x,n) - (";
    cout << GetK_value() <<", " << GetX_value() <<", " << GetN_value() << ") does
    not work!\n"; (move up a line)
    return false;
    }
}

//Function to calculate the sum of k left side
```

```
//Good!
inline unsigned long long int Individual::Summation_KLS(){
    sumK_LS = 0;
    for(unsigned int j = 1; j < GetK_value(); ++j){
        sumK_LS += j*j*j;
    }
    cout << "The summation of " << GetK_value() << " on the left-hand side is: "
     << sumK_LS << endl; (move up a line)
    return sumK_LS;
}

//Function to calculate x to thrid power
//Good!
inline unsigned long long int Individual::X_valueLS(){
    long double x_cubed(GetX_value()*GetX_value()*GetX_value());
    cout << GetX_value() << " raised to the third power on the left-hand side is: "
     << x_cubed << endl; (move up a line)
    return x_cubed;
}

//Function to calculate the sum of n left-hand side
//Good!
inline unsigned long long int Individual::Summation_NLS(){
    sumN_LS = 0;
    unsigned int begin(GetK_value() + 1);
    unsigned int condition(GetN_value() + 1);
    for(unsigned int j = begin; j < condition; ++j){
        sumN_LS += j*j*j;
    }
    cout << "The summation of " << GetN_value() << " on the left-hand side is: "
    << sumN_LS << endl; (move up a line)
    return sumN_LS;
```

```
}

//Function to calculate the sum of K right-hand side
//Good!
inline unsigned long long int Individual::Summation_KRS(){
    sumK_RS = 0;
    for(unsigned int j = 1; j < GetK_value(); ++j){
        sumK_RS += j;
    }
    cout << "The summation of " << GetK_value() << " on the right-hand side is: "
     << sumK_RS << endl; (move up a line)
    cout << "The value of x on the right hand side is: " << GetX_value() << endl;
    return sumK_RS;
}

//Function to calculate the sum of N right-hand Side
//Good!
inline unsigned long long int Individual::Summation_NRS(){
    sumN_RS = 0;
    unsigned int begin(GetK_value() + 1);
    unsigned int condition(GetN_value() + 1);
    for(unsigned int j = begin; j < condition; ++j){
        sumN_RS += j;
    }
    cout << "The summation of " << GetN_value() << " on the right-hand side is: "
     << sumN_RS << endl; (move up a line)
    return sumN_RS;
}

//Function to square right-hand side
//Good!
inline unsigned long long int Individual::Right_HSquare(){
```

```
    long double RHS(Summation_KRS() + GetX_value() + Summation_NRS());
    RHS *= RHS;
    cout << "The right-hand side summation squared is: " << RHS << endl;
    return RHS;
}
```

Individual.h Code For File:

```
//individual.h

/*Used to find the soltuions
  for user determined
  (k, x, n)
*/

#include "SumOfCubes.h"

#ifndef _INDIVIDUAL_H_
#define _INDIVIDUAL_H_

class Individual : public SumOfCubes{
    protected:

    public:
        Individual();
        void kSetter(int arr[])override;
        void xSetter(int arr[])override;
        void nSetter(int arr[])override;
        bool calc_NBoundaries()override;
        bool Boundaries()override;
        bool checkEquality()override;
        unsigned long long int Summation_NLS()override;
        unsigned long long int Summation_KLS()override;
        unsigned long long int Summation_KRS()override;
        unsigned long long int Summation_NRS()override;
        unsigned long long int X_valueLS()override;
        unsigned long long int Right_HSquare()override;
        void produceCombinations(int arr[], int data[], const int &,
```

```
        const int &, const int &, const int &)override; (move up a line)
};

#endif
```

NonTrivial.h Code For File:

```
//Non-Trivial.h

/*Used to find non-trivial
 solutions for user determined
 range for (k, x, n)
*/

#include "SumOfCubes.h"

#ifndef _NONTRIVIAL_H_
#define _NONTRIVIAL_H_

class NonTrivial : public SumOfCubes{
    private:
        int indexes;

    public:
        NonTrivial();
        void produceCombinations(int arr[], int data[], const int &,
         const int &, const int &, const int &)override; (move up a line)
        void kSetter(int arr[])override;
        void xSetter(int arr[])override;
        void nSetter(int arr[])override;
        int Amount();
        int Amount2();
        string Saved(int);
        int GetBeginRange();
        int GetEndRange();
        bool checkEquality()override;
        unsigned long long int Summation_NLS()override;
```

```
        unsigned long long int Summation_KLS()override;
        unsigned long long int Summation_KRS()override;
        unsigned long long int Summation_NRS()override;
        unsigned long long int X_valueLS()override;
        unsigned long long int Right_HSquare()override;
        bool Boundaries()override;
        bool calc_NBoundaries()override;
};

#endif
```

Nontrivial.cpp Code For File:

```
//NonTrivial.cpp
#include "NonTrivial.h"

NonTrivial::NonTrivial ()
:SumOfCubes(), indexes(0) {}

inline void NonTrivial::produceCombinations(int arr[], int data[], const int &start,
 const int &end, const int &index, const int &combinations){ (move up a line)
    //array to hold copy of range

    if(index == combinations){
        if(data[0] != data[1] && data[0] != data[2] && data[1] != data[2]){
            kSetter(data);
        }
    }

    // replace index with all possible
    // elements. The condition "end-i+1 >= r-index"
    // makes sure that including one element
    // at index will make a combination with
    // remaining elements at remaining positions

    for(unsigned int i = start; i <= end && (end - i + 1 >= combinations - index);
    ++i){ (move up a line)
        data[index] = arr[i];
        produceCombinations(arr, data, i+1, end, index+1, combinations);
    }
}

//Function to set k value
```

```
//Good!
inline void NonTrivial::kSetter(int arr[]) {
    this->k = arr[0];
    xSetter(arr);
}

inline void NonTrivial::xSetter(int arr[]){
    this->x = arr[1];
    nSetter(arr);
}

inline void NonTrivial::nSetter(int arr[]){
    this->n = arr[2];
    if(calc_NBoundaries() == true){
        if(Boundaries() == true){
            if(checkEquality() == true){
            }
        }
    }
}

//Function to test if numbers are correct
inline bool NonTrivial::Boundaries(){

    long double p(GetX_value() + GetK_value() - GetN_value());
        if(GetK_value() != GetN_value()){
            //Removed stipulation that x <= n for testing purposes
            if(GetX_value() != GetN_value()){ //&& GetX_value() <= GetN_value()){
                long double xPlus_K(GetX_value() + GetK_value());
                if(xPlus_K > GetN_value()){
                    if(GetK_value() >= 3){
                        if(GetX_value() >= 5){
```

```
                            long double nMinusOne(GetN_value() - 1);
                            if(GetK_value() != nMinusOne){
                                if( p >= 2){
                                    return true;
                                }
                            }
                        }
                    }
                }
            }
        return false;
}

inline bool NonTrivial::calc_NBoundaries(){
    int count(0);
    int q(GetN_value()*GetN_value() + GetN_value() + 1);
    if(q == 0){
        return false;
    }
    if(q == 3){
        return false;
    }
    if(q % 3 == 0){
        return false;
    }
    else{
        return true;
    }

    //     if(q % 7 == 0){
    //     return true;
```

```
    // }4
    // else{
    //     return false;
    // }
}

inline bool NonTrivial::checkEquality(){

    long double LHS(Summation_KLS() + X_valueLS() + Summation_NLS());
    if(LHS == Right_HSquare()){
        ListOfSolutions(GetK_value(), GetX_value(), GetN_value());
        return true;
    }
    return false;

}

//Function to calculate the sum of k left side
//Good!
inline unsigned long long int NonTrivial::Summation_KLS(){
    sumK_LS = 0;
    for(unsigned int j = 1; j < GetK_value(); ++j){
        sumK_LS += j*j*j;
    }
    return sumK_LS;
}

//Function to calculate x to thrid power
//Good!
inline unsigned long long int NonTrivial::X_valueLS(){
    long double x_cubed (GetX_value()*GetX_value()*GetX_value());
    return x_cubed;
```

```
}

//Function to calculate the sum of n left-hand side
//Good!
inline unsigned long long int NonTrivial::Summation_NLS(){
    sumN_LS = 0;
    unsigned int begin (GetK_value() + 1);
    unsigned int condition (GetN_value() + 1);
    for(unsigned int j = begin; j < condition; ++j){
        sumN_LS += j*j*j;
    }
    return sumN_LS;
}

//Function to calculate the sum of K right-hand side
//Good!
inline unsigned long long int NonTrivial::Summation_KRS(){
    sumK_RS = 0;
    for(unsigned int j = 1; j < GetK_value(); ++j){
        sumK_RS += j;
    }
    return sumK_RS;
}

//Function to calculate the sum of N right-hand Side
//Good!
inline unsigned long long int NonTrivial::Summation_NRS(){
    sumN_RS = 0;
    unsigned int begin(GetK_value() + 1);
    unsigned int condition(GetN_value() + 1);
    for(unsigned int j = begin; j < condition; ++j){
        sumN_RS += j;
```

```
    }
    return sumN_RS;
}

//Returns the size of the vector
int NonTrivial::Amount(){
    indexes = Answerlist.size();
    return indexes;
}

int NonTrivial::Amount2(){
    indexes = correctList.size();
    return indexes;
}
//Returns the results at each index
string NonTrivial::Saved(int index){
    return Answerlist.at(index);
}

int NonTrivial::GetBeginRange(){
    return this->beginRange;
}

int NonTrivial::GetEndRange(){
    return this->endRange;
}

//Function to square right-hand side
//Good!
unsigned long long int NonTrivial::Right_HSquare(){
    long double RHS(Summation_KRS() + GetX_value() + Summation_NRS());
    RHS *= RHS;
```

```
    return RHS;
}
```

SumOfCubes.h Code For File:

```
//SumOfCubes.h

//An upgraded version of this program will use pointers
//This will improve memory management

// #include "Buddy.h"
#include <iostream>
#include <array>
#include <fstream>
#include <string>
#include <sstream>
#include <vector>

#include "SubSet.h"
using namespace std;

#ifndef _SUMOFCUBES_H_
#define _SUMOFCUBES_H_


class SumOfCubes : public SubSet{
    protected:
        unsigned short int k;
        unsigned short int x;
        unsigned short int n;
        unsigned long long int sumK_LS;
        unsigned long long int sumN_LS;
        unsigned long long int sumK_RS;
        unsigned long long int sumN_RS;
        int beginRange;
```

```
    int endRange;
    int combinations;
    int SIZE_OF_ARRAY;
    vector<string> Answerlist;
    int pCount;
    int *range;
    int *rangeDuplicate;
    short int position;
    vector <SubSet> correctList;
    vector<int> PeeCount;
public:
    SumOfCubes();
    void DefineRange();
    virtual void produceCombinations(int arr[], int data[], const int &,
    const int &, const int &, const int &)=0;    (move up a line)
    virtual void kSetter(int arr[]) = 0;
    unsigned short int GetK_value();
    virtual void xSetter(int arr[]) = 0;
    unsigned short int GetX_value();
    virtual void nSetter(int arr[]) = 0;
    unsigned short int GetN_value();
    virtual bool Boundaries() = 0;
    virtual bool calc_NBoundaries() = 0;
    virtual bool checkEquality() = 0;
    virtual unsigned long long int Summation_NLS() = 0;
    virtual unsigned long long int Summation_KLS() = 0;
    virtual unsigned long long int Summation_KRS() = 0;
    virtual unsigned long long int Summation_NRS() = 0;
    virtual unsigned long long int X_valueLS()=0;
    virtual unsigned long long int Right_HSquare() = 0;
    void ListOfSolutions(const int &, const int &, const int &);
    void Print();
```

```
        void GetHighestPCount(unsigned int &);
        int GetPCount(unsigned int &i);
        void EmptyVector();
        void deletePointerArray();
        bool FindPattern(unsigned int &);
};

#endif
```

SumOfCubes.cpp Code For File:

```
//SumOfCubes.cpp

#include "SumOfCubes.h"

SumOfCubes::SumOfCubes()
: SubSet(0, 0, 0), k(0), x(0), n(0), sumK_LS(0),
sumN_LS(0), sumK_RS(0), sumN_RS(0),
beginRange(0), endRange(0), combinations(3),
SIZE_OF_ARRAY(0), Answerlist(0), pCount(2), range(nullptr),
rangeDuplicate(nullptr), position(0) {}

//Function to define range of numbers
//Good!
void SumOfCubes::DefineRange(){
    cout << "Please enter the Low-end of the range: ";
    cin >> this->beginRange;
    cout << endl;
    cout << "Please enter the high-end of the range: ";
    cin >> this->endRange;
    cout << endl;
    long double size = ((this->endRange - this->beginRange) + 1);
    SIZE_OF_ARRAY = size;
    range = new int[SIZE_OF_ARRAY];
    for(unsigned int i = 0; i < SIZE_OF_ARRAY; ++i){
        range[i] = this->beginRange;
        this->beginRange++;
    }
    cout << endl;
    rangeDuplicate = new int[combinations];
    size = SIZE_OF_ARRAY - 1;
```

```
    produceCombinations(range, rangeDuplicate, 0, size, 0, combinations);
}


//Function to return k value
//Good!
unsigned short int SumOfCubes::GetK_value(){
    return this->k;
}

//Function to return x value
//Good!
unsigned short int SumOfCubes::GetX_value(){
    return this->x;
}

//Function to get n value
//Good!
unsigned short int SumOfCubes::GetN_value(){
    return this->n;
}

//Function to store correct answers in vector
void SumOfCubes::ListOfSolutions(const int &k_value, const int &x_value, const int
&n_value){ (move up to the previous line)

    SubSet correct(k_value, x_value, n_value);
    correctList.push_back(correct);

}
```

```
void SumOfCubes::Print(){

    int iPCount;
    int jPCount;
    int location;
    int HighestPCount;
    int tabSpaces(1);

    ostringstream str1;
    string finalString;

    for(unsigned int i = 0; i < correctList.size(); ++i){
        GetHighestPCount(i);;
    }

    HighestPCount = pCount;
    pCount = 2;

    while(pCount < HighestPCount){
        cout << "When p = " << pCount << endl;
        str1 << "When p = " << pCount << endl;
        for(unsigned int i = 0; i < correctList.size(); ++i){
            if(GetPCount(i) == pCount){
                cout << "(" << correctList.at(i).KC << ", " << correctList.at(i).XC
                 << ", " << correctList.at(i).NC << ")" << "\t"; (move up a line)
                str1 << "(" << correctList.at(i).KC << ", " << correctList.at(i).XC
                << "," << correctList.at(i).NC << ")" << "\t"; (move up a line)
                if(tabSpaces % 5 == 4){
                    cout << endl;
                    str1 << endl;
                    ++tabSpaces;
                }
```

```
                    ++tabSpaces;
                }
            }
            ++pCount;
            tabSpaces = 1;
            cout << endl;
            str1 << endl;
        }

        finalString = str1.str();
        Answerlist.push_back(finalString);

}

inline void SumOfCubes::GetHighestPCount(unsigned int &i){

        int count = correctList.at(i).XC+correctList.at(i).KC - correctList.at(i). NC;
        if(count > pCount){
            pCount = count;
        }
}

inline int SumOfCubes::GetPCount(unsigned int &i){

        int count = correctList.at(i).XC+correctList.at(i).KC - correctList.at(i). NC;
        return count;
}

void SumOfCubes::EmptyVector(){
        pCount = 0;
        correctList.clear();
        Answerlist.clear();
```

```
}

void SumOfCubes::deletePointerArray(){
    delete [] range;
    range = nullptr;
    delete [] rangeDuplicate;
    rangeDuplicate = nullptr;
}

inline bool SumOfCubes::FindPattern(unsigned int &s){
    int checkN = correctList.at(s).NC;
    int theorem = (checkN*checkN+checkN + 1);
    if(theorem % 7 == 0){
        return true;
    }
    else{
        return false;
    }
}
```

Subset.h Code For File:

//SubSet.h

```
#include <vector>

#ifndef _SUBSET_H_
#define _SUBSET_H_

class SubSet{
    public:
        unsigned short int KC;
        unsigned short int XC;
        unsigned short int NC;
        SubSet(const int &, const int &, const int &);
};
#endif}
```

Subset.cpp Code For File:

```
//SubSet.cpp

#include "SubSet.h"

SubSet::SubSet(const int &KCorrect, const int &XCorrect, const int &NCorrect)
: KC(KCorrect), XC(XCorrect), NC(NCorrect) {}
```

Main.cpp Code For File:

```
//main.cpp
#include "SumOfCubes.h"
#include "Individual.h"
#include "Trivial.h"
#include "NonTrivial.h"
#include <fstream>
#include <cassert>

inline void WelcomeScreen(){
    cout << endl;
    cout << "Welcome to Sum of Cubes!\n";
    cout << "The purpose of this program is to\n";
    cout << "find combinations (k, x, n) that make\n";
    cout << "equations on the right and left equal\n";
    cout << endl;
    cout << "What would you like to do?\n";
    cout << endl;
}

inline void Menu(){
    cout <<"-------------------------------------------------------------------\n";
    cout << "1. Enter (k, x, n) individually\n";
    cout << "2. Find non-trivial solutions for (k, x, n) within a given range\n";
    cout << "3. Find Trivial solutions for (k, x, n) within a given range\n";
    cout << "4. Quit\n";
    cout << "-------------------------------------------------------------------\n";

}

int main(){
```

```
int option(0);
char choice;
int fileAppendage2(0);
int fileAppendage3(0);
int iterator(0);
int iterator2(0);
int Emptyarr[3];
string NonTrivialSolutions;
string TrivialSolutions;
string example;
ostringstream fileName2;
ostringstream fileName3;

Individual Individual;
NonTrivial NonTrivial;
Trivial Trivial;

ofstream outFS;

WelcomeScreen();
Menu();
cin >> option;
cout << endl;
while(option != 4){
    if(option == 1){
        Individual.kSetter(Emptyarr);
        Menu();
        cin >> option;
    }
    else
    if(option == 2){
```

```
    NonTrivial.DefineRange();
    NonTrivial.deletePointerArray();
    NonTrivial.Print();
    cout << endl;
    cout << "Would you like to save this data? (Y/N) ";
    cin >> choice;
    if(choice == 'Y' || choice == 'y'){
        fileName2 << "Non-Trivial Solutions" << fileAppendage2 << ".txt";
        NonTrivialSolutions = fileName2.str();
        outFS.open(NonTrivialSolutions);
        if(!outFS.is_open()){
            cout << "could not open file!\n";
        }
        else{
            iterator2 = NonTrivial.Amount2();
            iterator = NonTrivial.Amount();
            outFS << "The total number of solutions from " << NonTrivial.
            GetBeginRange() - NonTrivial.GetEndRange(); (move up a line)
            outFS << " to " << NonTrivial.GetEndRange() << " are: "
            << iterator2; (move up to previous line)
            outFS << endl;
            outFS << NonTrivial.Saved(0) << endl;
        }
    }
    ++fileAppendage2;
    fileName2.str(string());
    outFS.close();
    NonTrivial.EmptyVector();
    Menu();
    cin >> option;
}
else{
```

```
            Trivial.DefineRange();
            Trivial.deletePointerArray();
            Trivial.Print();
            cout << "Would you like to save this data? (Y/N) ";
            cin >> choice;
            if(choice == 'Y' || choice == 'y'){
                fileName3 << "Trivial Solutions" << fileAppendage3 << ".txt";
                TrivialSolutions = fileName3.str();
                outFS.open(TrivialSolutions);
                if(!outFS.is_open()){
                    cout << "could not open file!\n";
                }
                else{
                    iterator = Trivial.Amount();
                    outFS << "The total number of solutions from "
                    << Trivial.GetBeginRange(); (move up to previous line)
                    outFS << " to " << Trivial.GetEndRange() << "are: " << iterator;
                    outFS << endl;
                    for(unsigned int i = 0; i < iterator; ++i){
                        outFS << Trivial.Saved(i) << endl;
                    }
                }
            }
            ++fileAppendage3;
            fileName3.str(string());
            outFS.close();
            Trivial.EmptyVector();
            Menu();
            cin >> option;
        }
    }
```

```
    return 0;
}
```

Appendix C

How To Generate Trivial Solutions

How To Generate Trivial Solutions

This wasn't a focus in the research because it's relatively easier to generate than non-trivial solutions. However, for the overzealous mathematicians who would like to see what trivial solutions look like. I provided the C++ code below. Feel free to make comparisons between Non-trivial and Trivial solutions.

Trivial.cpp Code For File:

```
//Trivial.cpp
#include "Trivial.h"

//constructor
Trivial::Trivial()
: SumOfCubes(), indexes(0) {}

inline void Trivial::produceCombinations(int arr[], int data[], const int &start,
const int &end, const int &index, const int &combinations){ (move up a line)

    if(index == combinations){
        if(data[0] == data[1] || data[0] == data[2] || data[1] == data[2]){
            cout << endl;
            kSetter(data);
        }
    }

    // replace index with all possible
    // elements. The condition "end-i+1 >= r-index"
    // makes sure that including one element
    // at index will make a combination with
    // remaining elements at remaining positions

    for(unsigned int i = start; i <= end && (end - i + 1 >= combinations - index);
    ++i){ (move up a line)
        data[index] = arr[i];
        produceCombinations(arr, data, i+1, end, index+1, combinations);
    }
```

```
}

//Function to set k value
//Good!
inline void Trivial::kSetter(int arr[]){
    this->k = arr[0];
    xSetter(arr);
}

inline void Trivial::xSetter(int arr[]){
    this->x = arr[1];
    nSetter(arr);
}

inline void Trivial::nSetter(int arr[]){
    this->n = arr[2];
    if(calc_NBoundaries() == true){
        if(Boundaries() == true){
            if(checkEquality() == true){
            }
        }
    }
}

//Function to test if numbers are correct
inline bool Trivial::Boundaries(){

    long double p = (GetX_value() + GetK_value() - GetN_value());
                long double xPlus_K = GetX_value() + GetK_value();
                if(xPlus_K > GetN_value()){
                    if(GetK_value() >= 3){
                        if(GetX_value() >= 5){
```

```
                            long double nMinusOne = GetN_value() - 1;
                            if(GetK_value() != nMinusOne){
                                if( p >= 2){
                                    return true;
                                }
                            }
                        }
                    }
                }
    return false;
}

inline bool Trivial::calc_NBoundaries(){
    int count = 0;
    int q = (GetN_value()*GetN_value() + GetN_value() + 1);
    if(q == 0){
        return false;
    }
    if(q == 3){
        return false;
    }
    if(q % 3 == 0){
        return false;
    }
    else{
        return true;
    }
}

inline bool Trivial::checkEquality(){

    long double LHS = (Summation_KLS() + X_valueLS() + Summation_NLS());
```

```
    if(LHS == Right_HSquare()){
        ListOfSolutions(GetK_value(), GetX_value(), GetN_value());
        return true;
    }
    return false;

}

//Function to calculate the sum of k left side
//Good!
inline unsigned long long int Trivial::Summation_KLS(){
    sumK_LS = 0;
    for(unsigned int j = 1; j < GetK_value(); ++j){
        sumK_LS += j*j*j;
    }
    return sumK_LS;
}

//Function to calculate x to thrid power
//Good!
inline unsigned long long int Trivial::X_valueLS(){
    long double x_cubed = (GetX_value()*GetX_value()*GetX_value());
    return x_cubed;
}

//Function to calculate the sum of n left-hand side
//Good!
inline unsigned long long int Trivial::Summation_NLS(){
    sumN_LS = 0;
    unsigned int begin = GetK_value() + 1;
    unsigned int condition = GetN_value() + 1;
    for(unsigned int j = begin; j < condition; ++j){
```

```
        sumN_LS += j*j*j;
    }
    return sumN_LS;
}

//Function to calculate the sum of K right-hand side
//Good!
inline unsigned long long int Trivial::Summation_KRS(){
    sumK_RS = 0;
    for(unsigned int j = 1; j < GetK_value(); ++j){
        sumK_RS += j;
    }
    return sumK_RS;
}

//Function to calculate the sum of N right-hand Side
//Good!
inline unsigned long long int Trivial::Summation_NRS(){
    sumN_RS = 0;
    unsigned int begin = GetK_value() + 1;
    unsigned int condition = GetN_value() + 1;
    for(unsigned int j = begin; j < condition; ++j){
        sumN_RS += j;
    }
    return sumN_RS;
}

//Function to square right-hand side
//Good!
inline unsigned long long int Trivial::Right_HSquare(){
    long double RHS = (Summation_KRS() + GetX_value() + Summation_NRS());
    RHS *= RHS;
```

```
    return RHS;
}

//Returns the size of the vector
int Trivial::Amount(){
    indexes = Answerlist.size();
    return indexes;
}

string Trivial::Saved(int index){
    return Answerlist.at(index);
}

int Trivial::GetBeginRange(){
    return this->beginRange;
}
int Trivial::GetEndRange(){
    return this->endRange;
}
```

Trivial.h Code For File:

```
//SumOfCubes2.h
#include "SumOfCubes.h"

#ifndef _TRIVIAL_H_
#define _TRIVIAL_H_

class Trivial : public SumOfCubes{
    private:
        int indexes;
    public:
        Trivial();
        void produceCombinations(int arr[], int data[], const int &,
        const int &, const int &, const int &)override; (move up a line)
        int Amount();
        string Saved(int);
        void kSetter(int arr[])override;
        void xSetter(int arr[])override;
        void nSetter(int arr[])override;
        int GetBeginRange();
        int GetEndRange();
        bool checkEquality()override;
        unsigned long long int Summation_NLS()override;
        unsigned long long int Summation_KLS()override;
        unsigned long long int Summation_KRS()override;
        unsigned long long int Summation_NRS()override;
        unsigned long long int X_valueLS()override;
        unsigned long long int  Right_HSquare()override;
        bool Boundaries()override;
        bool calc_NBoundaries()override;
```

```
};
#endif
```

Bibliography

[Apo08] T.A. Apostol. *Primer on Bernoulli Numbers and Polynomials.* Mathematics Magazine, 81(3), 178-190. 2008. Retrieved from www.jstor.org/stable/27643104

[Bea96] A.F. Beardon.*Sums of Powers of Integers*, American Mathematical Monthly. pp. 201-213. 1996.

[Rob07] Robert, Booth, and Hieu D. Nguyen. *Bernoulli Polynomials and Pascal's Square.* Rowan University, Department of Mathematics, pp. 1-12, 26 June 2007.

[Bur98] D. M. Burton. *Elementary Number Theory.* The McGraw-Hill Companies, Inc.1998.

[Das11] Mohammad Torabi Dashti. *Faulhaber's Triangle.* The College of Mathematics Journal, Taylor & Amp; Francis, Ltd. On Behalf of the Mathematical Associa tion of America, Vol. 42, No. 2, pp. 16 - 97, Mar. 2011.

[Don93] Donald E. Knuth. *Johann Faulhaber and sums of powers.* Mathematics of Com putation 61 (203) pp. 277-294, 1993.

[Fau20] *Faulhaber's Triangle.* Faulhaber's Triangle - Rosetta Code, rosettacode.org/wiki/ Faulhaber's triangle, 30 May 2020.

[Fej20] Hajrudin Fejzić. *A Note On Faulhaber's Polynomials*, 2020.

[Fej05] H. Fejzić, D. Rinne, B. Stein.*On Sums of Cubes.* The College Mathematics Journal, 36(3), 226-228. doi:10.2307/30044858, 2005.

[Fre90] French, D. *Sums of Squares and Cubes.* Mathematics in School, 19(3), 34-37. Retrieved from www.jstor.org/stable/30214681, 1990.

[Hai18] P. Haile. *Differentiating an Integral.* Yale, www.econ.yale.edu/ pah29/409web/leibniz. pdf. Sept, 2018.

[Haj20] Hajrudin Fejzić. *Finding coefficients of Faulhaber's polynomials*, 2020.

[Joh31] Johann Faulhaber. *Academia Algebrae - Darinnen die miraculosische Inven tiones zu den höchsten Cossen weiters continuirt und profitiert werden* (1631).

[Jac34] Carl Jacobi. *De usu legitimo formulae summatoriae Maclaurinianae*, Journal für die reine und angewandte Mathematik 12. pp. 263-272, 1834.

[Lar19] Nathaniel Larson. *The Bernoulli Numbers: A Brief Primer.* Whitman. www.whit man.edu/Documents/Academics/Mathematics/2019/Larson-Balof.pdf. 10 May 2019.

[Mar70] Ash J. Marshall. *A Characterization of the Peano Derivative.* Depaul Univer sity, Transactions of the American Mathematical Society, Vol.149, pp. 489 - 501. June, 1970. https://math.depaul.edu/mash/Peano.pdf.

[Mer91] Uta C. Merzbach and Carl B. Boyer. *A History of Mathematics.* John Wi ley & Amp; Sons, 1991.

[Owe92] R. Owens.*Sums of Powers of Integers.* Mathematics Magazine, 65(1), 38-40. doi:10.2307/2691359, 1992.

[Pen19] David Pengelley. *Figurate Numbers and Sums of Numerical Powers: Fermat, Pascal, Bernoulli.* MAA. pp. 1-29. Retrieved December 19, 2019.

[Pol09] George Pólya. *Mathematical Discovery on Understanding, Learning, and Teach ing Problem Solving.* Ishi Press, 2009.

[Rik10] Tom Rike. *Sums of Powers and Bernoulli Numbers.* Math Circle, Berkley Math Circle, 30 Mar. 2010.

[Sum19] *Sum of n, n^2, or n^3.* Brilliant Math & amp; Science Wiki, brilliant.org/wiki/sum-of-n-n2-or-n3/. Retrieved December 16, 2019.

[Sum20] *Sums of Powers.* Math Pages. mathpages.com/home/kmath279/kmath279.htm. Retrieved June 18, 2020.

[Tay20] *Taylor formula.* Encyclopedia of Mathematics. URL: http://www.encyclopediaofmath.org/index.php? title=Taylor formula&oldid=31209. Retrieved March 12, 2020.

[Tom16] Tom N. Chu. *Summation Formulas Involving Polynomials.* The Mathematics Teacher, 109(5), 393-397. doi:10.5951/mathteacher.109.5.0393, 2016.

[Wel19] Karolee Weller. *Carl Friedrich Gauss 1777-1855.* Gauss, Wichita. www.math.wichita.edu/history/men/gauss.gtml. Retrieved December 20,2019.