PROOF THAT THE EQUATION $A! \times B! = C!$ HAS ONLY ONE SOLUTION FOR INTEGERS $1 < A < B < C - 1$

Ken Clements
ken-clements@ieee.org

Received: ___________, *Revised:* ___________, *Accepted:* ___________, *Published:* ___________

Abstract

Here it is proved that the Diophantine equation $a! \times b! = c!$ has only one solution in positive integers a, b, c satisfying $1 < a < b < c - 1$, specifically $(a, b, c) = (6, 7, 10)$. The proof leverages prime factorization properties, and requirements that no prime factor be missing, to establish the uniqueness of the solution. Closures are also proved for primorials as consecutive products of integers, and for solutions to Brocard's equation.

1. Introduction

The aim of this presentation is to prove that the Diophantine equation:

$$a! \times b! = c!$$

has only one solution in positive integers a, b, c satisfying $1 < a < b < c - 1$, specifically:

$$(a, b, c) = (6, 7, 10).$$

This equation has intrigued mathematicians due to its connections with factorial numbers and prime factorizations.[4] It this proof, it will be demonstrated that the only solution in positive integers a, b, c with $1 < a < b < c-1$ is $(a, b, c) = (6, 7, 10)$.

The approach to be shown, involves analyzing the prime factorizations of factorials, which have no missing primes (the "π-complete" property), and checking the quotient $\frac{c!}{b!}$ for that property. This will be accomplished by (1) showing that the equation can be analyzed as the products of k consecutive integers (k-tuples); (2) showing that those k-tuple products must not be missing prime factors in order to equal factorials; (3) showing that for sufficiently large c they must be missing prime factors; (4) showing that below that sufficiently large c, a computer search eliminates all possible solutions except $(a, b, c) = (6, 7, 10)$.

2. Definitions

Definition 1 (*Factorial Function)***.** For a positive integer n, the factorial $n!$ is defined as

$$n! = n \times (n-1) \times \cdots \times 1.$$

Definition 2 (*ABC Factorial Equation)***.** For the purpose of this proof, define the *ABC Factorial Equation* as the Diophantine equation $a! \times b! = c!$.

Definition 3 (*a,b,c)***.** For the purpose of this proof, define a, b, and c as integer values sought for solution of the ABC Factorial Equation where $1 < a < b < c - 1$.

Definition 4 (*k)***.** For the purpose of this proof, define $k = c - b$.

Definition 5 (*K-tuple Product)***.** For the purpose of this proof, define $P_k(c) = c \times (c-1) \times \cdots \times (c-k+1)$.

Definition 6 (*ABC Factorial Conjecture)***.** For the purpose of this proof, define the *ABC Factorial Conjecture* to be the proposition that the ABC Factorial Equation has only a single non-trivial solution at $(a, b, c) = (6, 7, 10)$.

Definition 7 (*Primorial Function)***.** For a positive integer r, the primorial number, $r\#$, is defined as

$$r\# = \prod_{i=1}^{r} p_i$$

where p_i is the i-th prime number.

Definition 8 (*Prime Factorization)***.** By the fundamental theorem of arithmetic[1,17], every integer $n > 1$ can be uniquely represented as a product of prime powers:

$$n = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r},$$

where p_i are prime numbers in increasing order, and e_i are non-negative integers. This will be referred to as the *prime factorization* of the integer n.

Definition 9 (*Factorial Exponent Form (FEF)*)**.** An integer n has the *Factorial Exponent Form* property if, and only if, in its prime factorization, the exponent of its largest prime factor $p_{\max}$ is exactly 1, and all exponents e_i for primes from 2 (where $i = 1$) to $p_i \leq p_{\max}$ are monotonically non-increasing, that is, $e_{i+1} \leq e_i$.

Definition 10 (*Primorial Complete (π-complete) Property*)**.** An integer $n > 1$ has the *Primorial Complete* property if, and only if, in its prime factorization, all exponents e_i for primes $p_i \leq p_{\max}$ are at least 1, where $p_{\max}$ is the largest prime factor of n. That is, there are no missing primes up to $p_{\max}$.

Definition 11 (*π-complete Numbers*)**.** An integer $n > 1$ is *π-complete* if, and only if, it has the Primorial Complete property.

Definition 12 (*π-complete Order*)**.** A *π-complete Order*, r, is the set of all π-complete numbers that can be expressed as:

$$r\#m = r\# \times m$$

where $r\#$ is the r-th primorial number and m is a positive integer having no prime factor larger than the largest prime factor of $r\#$, p_r. This set is denoted as $\mathcal{P}_c(r)$, and the set of all π-complete numbers is denoted $\mathbb{P}_c$ where $\mathbb{P}_c \subseteq \mathbb{N}$ and

$$\mathbb{P}_c := \bigcup_{r=1}^{\infty} \mathcal{P}_c(r).$$

Definition 13 (P_{cm})**.** For any set of integers $n > 1$ the P_{cm} of that set is largest value in the set that has the π-complete property. $P_{\mathrm{cm}} = 0$ if the set is empty or has no values with the π-complete property.

Definition 14 (*Factor Delta Function*)**.** For any positive integer, n, with positive integers $d \leq e$, such that $n = d \times e$, and $e - d$ is the smallest difference for any two integers with product equal to n, then the function

$$\mathcal{F}_\delta(n) = (e - d)$$

is defined to be this minimum difference.

3. Lemmas

Lemma 1 (*Exponents of Primes in Factorials*)**.** *The exponent of a prime p in the prime factorization of $n!$ is given by:*

$$\nu_p(n!) = \sum_{i=1}^{\infty} \left\lfloor \frac{n}{p^i} \right\rfloor$$

where $\nu_p(n!)$ denotes the exponent of p in $n!$, and $\lfloor x \rfloor$ is the floor function.

Proof. Proved by Legendre.[7] □

Lemma 2 (*Largest Prime in Factorial*)**.** *For any integer $n > 1$, in the prime factorization of $n!$, the largest prime factor $p \leq n$ appears with exponent exactly 1.*

Proof. Let p be the largest prime number less than or equal to n, then:

1. First, let it be proven that p appears at least once in $n!$:

- Since $p \leq n$, p is one of the factors in the product $n!$.
- Therefore, p appears at least once in the prime factorization of $n!$.

2. Now, let it be proven that p cannot appear more than once:

- By Legendre's Theorem, the exponent of p in $n!$ is given by:

$$\nu_p(n!) = \sum_{k=1}^{\infty} \left\lfloor \frac{n}{p^k} \right\rfloor.$$

- Since p is the largest prime $\leq n$, it follows, from Chebyshev's Theorem[1,75], that $n < 2p$.
- Therefore:

$$\left\lfloor \frac{n}{p} \right\rfloor = 1$$

  and

$$\left\lfloor \frac{n}{p^2} \right\rfloor = 0.$$

- All higher terms in the sum are also zero.
- Thus, $\nu_p(n!) = 1$.

Therefore, p appears exactly once in the prime factorization of $n!$. □

Corollary 1 (*No Factorial Perfect Power)***.** *For any integer $n > 1$, $n!$ cannot be a perfect power.*

Proof. Let $n > 1$ be any integer. By Lemma 2, the largest prime $p \leq n$ appears in the prime factorization of $n!$ with exponent exactly 1.

Suppose, for contradiction, that $n!$ is a perfect power. Then $n! = m^j$ for some integers m, j where $j > 1$.

This means that every prime factor in the prime factorization of $n!$ must have an exponent that is divisible by j. However, the largest prime factor has exponent 1, which is not divisible by any integer $j > 1$.

This contradiction proves that $n!$ cannot be a perfect power. □

Lemma 3 (*Behavior of Exponents in $c!/b!$)***.** *For integers c and b with $c > b$, the exponents of primes in $c!/b!$ are given by the differences:*

$$\nu_p\left(\frac{c!}{b!}\right) = \nu_p(c!) - \nu_p(b!) = \sum_{i=1}^{\infty}\left(\left\lfloor \frac{c}{p^i}\right\rfloor - \left\lfloor \frac{b}{p^i}\right\rfloor\right).$$

Proof. Let p be any prime number, then:

1. By Legendre's formula[7], for any positive integer n, the exponent of p in $n!$ is:

$$\nu_p(n!) = \sum_{i=1}^{\infty}\left\lfloor \frac{n}{p^i}\right\rfloor.$$

2. For the quotient $c!/b!$, the exponent of p is the difference of the exponents:

$$\nu_p\left(\frac{c!}{b!}\right) = \nu_p(c!) - \nu_p(b!).$$

3. Substituting Legendre's formula for both terms:

$$\nu_p\left(\frac{c!}{b!}\right) = \sum_{i=1}^{\infty}\left\lfloor \frac{c}{p^i}\right\rfloor - \sum_{i=1}^{\infty}\left\lfloor \frac{b}{p^i}\right\rfloor.$$

4. Since $c > b$, both sums are finite (terminating when $p^i > c$), allowing combination:

$$\nu_p\left(\frac{c!}{b!}\right) = \sum_{i=1}^{\infty}\left(\left\lfloor \frac{c}{p^i}\right\rfloor - \left\lfloor \frac{b}{p^i}\right\rfloor\right).$$

□

Lemma 4 (*Monotonic Factorial Exponents*)**.** *For any positive integer n, in the prime factorization of $n!$, the exponents of consecutive prime factors form a monotonically non-increasing sequence. That is, if:*

$$n! = 2^{e_1}3^{e_2}5^{e_3}\cdots p_r^{e_r}$$

where p_r is the largest prime $\leq n$, then:

$$e_1 \geq e_2 \geq e_3 \geq \cdots \geq e_r = 1.$$

Proof. Let p_i be the i-th prime number. For any prime $p_i \leq n$, its exponent e_i in $n!$ is given by:

$$\alpha_i = \sum_{j=1}^{\infty} \left\lfloor \frac{n}{p_i^j} \right\rfloor .$$

For consecutive primes p_i and p_{i+1}:

1. For each power j,
$$\left\lfloor \frac{n}{p_i^j} \right\rfloor \geq \left\lfloor \frac{n}{p_{i+1}^j} \right\rfloor$$
since $p_i < p_{i+1}$.

2. The sum for each prime terminates when $p^j > n$.

3. The sum for p_{i+1} has fewer non-zero terms than the sum for p_i.

Therefore, $e_i \geq e_{i+1}$ for all i, and by Lemma 2, $e_r = 1$ for the largest prime $p_r \leq n$. □

Lemma 5 (*Factorials have FEF*)**.** *For any integer $n > 1$, $n!$ has the FEF property.*

Proof. Let $n > 1$ be any integer. It will be demonstrated that $n!$ satisfies both requirements of the FEF property:

1. First, it can be shown that the exponents form a monotonically non-increasing sequence: For any prime p_i, its exponent α_i in $n!$ is given by Legendre's formula:
$$\alpha_i = \sum_{j=1}^{\infty} \left\lfloor \frac{n}{p_i^j} \right\rfloor .$$
For consecutive primes $p_i < p_{i+1}$, it follows that:
$$\left\lfloor \frac{n}{p_i^j} \right\rfloor \geq \left\lfloor \frac{n}{p_{i+1}^j} \right\rfloor$$

for all $j \geq 1$, since $p_i < p_{i+1}$. Therefore, summing over all j:

$$\alpha_i = \sum_{j=1}^{\infty} \left\lfloor \frac{n}{p_i^j} \right\rfloor \geq \sum_{j=1}^{\infty} \left\lfloor \frac{n}{p_{i+1}^j} \right\rfloor = \alpha_{i+1}$$

.

2. The largest prime factor has exponent exactly 1 by Lemma 2.

Therefore, $n!$ has monotonically non-increasing exponents ending in 1, satisfying the FEF property. □

Corollary 2 (*FEF Necessary for Factorial*)**.** *If a number $n > 1$ does not have the FEF property, then n is not a factorial number.*

Proof. The corollary is the contrapositive of Lemma 5 □

Lemma 6 (*FEF implies π-complete*)**.** *If an integer $n > 1$ has the FEF property, then n also has the π-complete property.*

Proof. Let $n > 1$ be a number with the FEF property. Consider its prime factorization:

$$n = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$$

where $p_1 < p_2 < \cdots < p_r$ are primes and e_i are their exponents.

By definition of the FEF property:

1. The exponents form a monotonically decreasing sequence: $e_1 \geq e_2 \geq \cdots \geq e_r$.

2. The last exponent is 1: $e_r = 1$.

Therefore:

1. Since $e_r = 1$ and the sequence is monotonically non-increasing, $e_i \geq 1$ for all i.

2. This means no exponent can be zero.

Thus, n has no zero exponents in its prime factorization up to its largest prime factor p_r, which is the definition of the π-complete property. □

The next lemma will provide the main working tool for the overall proof. It simply states that because all factorial numbers, $n!$, have every prime factor $\leq n$, if a given number is missing one or more prime factors in its prime factorization below its largest prime factor, then it cannot be a factorial number.

Lemma 7 (*π-complete Necessary for Factorial*)**.** *If an integer greater than one does not have the π-complete property, then it is not equal to any $n!$.*

Proof. By Corollary 2 (FEF Necessary for Factorial) $n!$ must have FEF. Then by Lemma 6 (FEF implies π-complete) $n!$ must be π-complete. Thus by the contrapositive of that lemma, if a number greater than one is not π-complete, then it is not equal to any $n!$. □

Lemma 8 (*π-complete Multiplication*)**.** *Given π-complete numbers M and N, the product $M \times N$ is a π-complete number.*

Proof.

1. If $M = N$, then the prime factorization of their product is simply the same as M and N, except that each prime factor will have its exponent doubled. No prime factors will be lost, so the product will be π-complete.
2. If $M \neq N$ but they have the same number of prime factors, then their product will have a prime factorization with exponents comprising the sum of the prime factorization exponents of M and N, so no prime factors will be lost, and the product will be π-complete.
3. If M and N have differing numbers of prime factors, then because they are both π-complete, one must have all the prime factors of the other. This means that their product will have all the prime factors of the number with the larger number of prime factors, and no prime factors will be lost, so the product will be π-complete.

□

Lemma 9 (*π-complete and N Multiplication*)**.** *Given a π-complete number, M, and a positive integer, N, such that N has no prime factor greater than the greatest prime factor of M, then the product $M \times N$ is π-complete.*

Proof. Because N has no prime factor greater than the greatest prime factor of M, and M has that prime factor, and all prime factors less than that prime factor (M being π-complete), the product $M \times N$ will have the same complete set of prime factors as does M, with exponents increased by the corresponding exponents of the prime factors of N, and so the product will be π-complete. □

Lemma 10 (*Primorial-Smooth Representation*)**.** *Every π-complete number, N, can be represented as $r\#m$ where:*

- *r is the number of the greatest prime factor of N, p_r.*

- *$r\#$ is the primorial of order r, that is, the product of all primes up to and including the r-th prime number, p_r.*
- *m is a positive integer whose prime factors are all less than or equal to the r-th prime number, that is, m is p_r-smooth.*
- *$r\#m = r\# \times m$.*

Proof. By definition, the prime factorization of $r\#$ has all ones for the prime factor exponents up to the greatest prime factor, p_r. Because N is π-complete, it also has all those same prime factors up to p_r. Let m be the quotient $\frac{N}{r\#}$. Because N is π-complete, it has prime factors with exponents of at least 1 for every prime number from 2 to p_r, so subtracting 1 from each of those exponents will not result in an exponent less than zero for any prime factor in $m = \frac{N}{r\#}$. Therefore, $r\#$ divides N, and N can be represented as $r\# \times m$ or $r\#m$. □

Lemma 11 (*Disjointness of π-complete Orders*)**.** *Let $\mathcal{P}_c(r)$ be the π-**complete order** of rank r, consisting of all π-complete integers whose largest prime factor is the r-th prime p_r. Then for any two distinct positive integers r and s,*

$$\mathcal{P}_c(r) \cap \mathcal{P}_c(s) \; = \; \varnothing.$$

In other words, a π-complete integer cannot lie in more than one π-complete order.

Proof. By definition of the π-complete order $\mathcal{P}_c(r)$, every integer $n \in \mathcal{P}_c(r)$ has its largest prime factor p_r. Concretely, if

$$n \; = \; 2^{e_1} \times 3^{e_2} \times \cdots \times p_r^{e_r} \quad (e_i > 0),$$

then p_r is the maximal prime appearing in the factorization of n.

Suppose, for contradiction, that some integer n belongs to two different orders, $\mathcal{P}_c(r)$ and $\mathcal{P}_c(s)$, with $r \neq s$. Then n must simultaneously have largest prime factor p_r *and* largest prime factor p_s. Since p_r and p_s are distinct primes, it is impossible for a single integer to have two different "largest primes." This contradiction establishes that no integer can lie in both $\mathcal{P}_c(r)$ and $\mathcal{P}_c(s)$ when $r \neq s$.

Hence the sets $\mathcal{P}_c(r)$ and $\mathcal{P}_c(s)$ are pairwise disjoint whenever $r \neq s$ for all $\mathbb{P}$. □

Lemma 12 (*Larger Factorial Prime*)**.** *For all integers $a > 1$ and $b > 1$, if $a!$ has a prime factor p_a that is larger than any prime factor in $b!$, then $a > b$.*

Proof. Proof is by contradiction. Assume $a \leq b$ and $a!$ has a prime factor p_a larger than any prime factor in $b!$.

1. Since $a!$ has prime factor p_a, there must be an integer $j \leq a$ such that:

   (a) Either $j = p_a$
   (b) Or j is composite and has p_a as a prime factor.

2. In either case, $j \leq a \leq b$

3. Therefore j is one of the factors in $b!$ since $b!$ includes all integers from 1 to b

4. This means that $b!$ must also have p_a as a prime factor because:

   (a) If $j = p_a$, then p_a is directly a factor of $b!$.
   (b) If j is composite containing p_a, then p_a appears in the prime factorization of $b!$.

5. This contradicts the assumption that p_a is larger than any prime factor in $b!$.

Therefore, $a > b$ must be true. □

Lemma 13 (*Factorial π-complete Form*)**.** *For any integer $n > 1$, let r be the number of primes less than or equal to n. Then $n!$ can be written as $r\#m$ where m is the product of all composite numbers less than or equal to n.*

Proof. Let n be an integer greater than 1 and r be the number of primes less than or equal to n.

1. First, observe that $n!$ can be written as:
$$n! = (r\#) \cdot \prod_{j \leq n, j \text{ composite}} j.$$

2. Let $m = \prod_{j \leq n, j \text{ composite}} j$. It then needs to be shown that m is p_r-smooth.

3. Consider any composite number $j \leq n$. By definition:

   (a) j can be factored into primes: $j = q_1^{e_1} q_2^{e_2} \cdots q_l^{e_l}$.
   (b) Since $j \leq n$, each prime factor q_i must be less than or equal to n.
   (c) Therefore, each q_i is less than or equal to the largest prime $\leq n$.

4. Since r is the count of primes up to n:

   (a) The r-th prime is the largest prime $\leq n$.
   (b) All prime factors of any composite $j \leq n$ must be $\leq$ r-th prime.
   (c) Therefore, m is p_r-smooth.

Thus, $n! = r\#m$ where m is p_r-smooth, having prime factors all less than or equal to the r-th prime. □

Lemma 14 (*Factorial Inequality*)**.** *For any integers $a > 1$ and $b > 1$, if $a! > b!$, then $a > b$.*

Proof. This is proven by contradiction. Suppose $a! > b!$ and $a < b$. Then:

$$b! = a! \times \prod_{i=a+1}^{b} i.$$

Since all factors in the product are positive integers:

$$b! > a!.$$

This contradicts the initial assumption that $a! > b!$. Therefore, if $a! > b!$, then $a > b$. □

In the main proof, the search for a candidate $a!$ will proceed by examination of products of consecutive integers. The next lemma sets the stage for that.

Lemma 15 (*Quotient of Factorials as Consecutive Integer Products*)**.** *Let b and c be integers such that $b > 1$ and $c > b$. Define $k = c - b$. Then $b!$ divides $c!$, and the resulting quotient is the product of k consecutive integers:*

$$\frac{c!}{b!} \;=\; c \times (c-1) \times \cdots \times \bigl(b+1\bigr),$$

which can equivalently be written as

$$P_k(c) \;=\; \prod_{\ell=0}^{k-1} \bigl(c-\ell\bigr).$$

Proof. By definition,

$$c! \;=\; c \times (c-1) \times (c-2) \times \cdots \times (b+1) \;\times\; b \times (b-1) \times \cdots \times 1.$$

Since $b!$ is precisely

$$b! \;=\; b \times (b-1) \times \cdots \times 1,$$

all the factors $b, (b-1), \ldots, 1$ in $c!$ can be canceled out by dividing by $b!$. Thus,

$$\frac{c!}{b!} \;=\; c \;\times\; (c-1) \;\times \cdots \times (b+1).$$

The integer $k = c - b$ then counts exactly how many factors appear in this product: it runs from c downward to $(b+1)$, inclusive. Hence this is the product of the k consecutive integers

$$c,\; c-1,\; \ldots,\; b+1.$$

This proves that $b!$ divides $c!$ and that the quotient is the product of exactly k consecutive descending integers starting at c. □

Lemma 16 (*Largest Prime in $c!$ Must Appear in $b!$*)**.** *Let $1 < a < b < c - 1$ be positive integers satisfying*

$$a! \times b! \;=\; c!.$$

Denote by p_c the largest prime factor of $c!$. Then p_c must also appear in the prime factorization of $b!$ (with exponent exactly 1).

Proof. By Lemma 2 (Largest Prime in Factorial), the largest prime factor, p_c, in $c!$ appears exactly once in the factorization of $c!$.

Suppose, for contradiction, that p_c does *not* appear in $b!$. Then, when forming the quotient

$$\frac{c!}{b!} \;=\; a!,$$

the prime p_c remains in the factorization of $a!$. But this implies $a!$ contains a prime factor p_c that is larger than *any* prime factor of $b!$. By Lemma 12 (Larger Factorial Prime), that can only happen if $a > b$. This contradicts our assumption that $1 < a < b$.

Therefore, the largest prime factor p_c in $c!$ must indeed occur in $b!$ as well. Since the exponent of p_c in $c!$ is 1, $b!$ must also carry that prime to the exact same exponent, with exponent $= 1$. Hence the lemma is proved. □

The following lemma is a very important limitation on how much lower the value of b can be, with regard to the value of c, and still have a possible $a! \times b! = c!$.

Lemma 17 (*Limiting k by $c - p_c$*)**.** *Suppose $1 < a < b < c - 1$ are positive integers satisfying*

$$a! \;\times\; b! \;=\; c!,$$

and let $k = c - b$. Let p_c be the largest prime with $p_c < c$. Then

$$k \;\leq\; c \;-\; p_c.$$

Equivalently, $b \;\geq\; p_c$.

Proof. Assume for contradiction that $k > c - p_c$. Then

$$k = c - b \;>\; c - p_c \quad \text{implies} \quad b < p_c..$$

Since p_c is the largest prime less than c, p_c appears exactly once in $c!$ by Lemma 2. Because $b < p_c$, p_c does *not* appear in $b!$. Consequently, when forming the quotient

$$\frac{c!}{b!} \;=\; a!,$$

the prime p_c must appear fully in $a!$. But this implies $a!$ has a prime factor p_c that is larger than any prime in $b!$. By Lemma 12 (Larger Factorial Prime), this implies $a > b$, contradicting $1 < a < b$.

Therefore, the assumption that $k > c - p_c$ is false; hence $k \leq c - p_c$. Equivalently, $b \geq p_c$. □

The following lemma sets a limit on how far the geometric mean, $\sqrt{n(n-1)}$, of two consecutive integers can be below the arithmetic mean, $\frac{n+(n-1)}{2}$, of those two integers. This lemma will be used many times in the subsequent proof to show that as the products of consecutive integers becomes larger, fewer and fewer of those products can have a contiguous set of prime factors all the way down to 2.

Lemma 18 (*Square Root Limitation*). *For all integers $n > 1$, the following inequality holds:*

$$\frac{2n-1}{2} - \sqrt{n(n-1)} < \frac{1}{n-1}$$

Proof. This will be proven by contradiction. Assume there exists some $n > 1$ where:

$$\frac{2n-1}{2} - \sqrt{n(n-1)} \geq \frac{1}{n-1}$$

With $n > 1$ both sides are greater than zero, so rearrange and square both sides to:

1. Start with:

$$\left(\frac{2n-1}{2} - \frac{1}{n-1}\right)^2 \geq n(n-1).$$

2. Multiply both sides by $(n-1)^2$:

$$\left(\frac{(2n-1)(n-1)}{2} - 1\right)^2 \geq n(n-1)^3.$$

3. Expand $(2n-1)(n-1)$:

$$\left(\frac{2n^2-2n-n+1}{2} - 1\right)^2 \geq n(n-1)^3.$$

4. Simplify numerator:

$$\left(\frac{2n^2-3n+1}{2} - 1\right)^2 \geq n(n-1)^3.$$

5. Combine terms:

$$\left(\frac{2n^2-3n-1}{2}\right)^2 \geq n(n-1)^3.$$

6. Multiply out $(n-1)^3$:

$$\left(\frac{2n^2-3n-1}{2}\right)^2 \geq n(n^3-3n^2+3n-1).$$

7. Expand left side:

$$\frac{4n^4-12n^3+9n^2+4n^2-6n+1}{4} \geq n^4-3n^3+3n^2-n.$$

8. Combine like terms:

$$4n^4-12n^3+13n^2-6n+1 \geq 4n^4-12n^3+12n^2-4n.$$

9. Subtract right side from both sides:

$$n^2-2n+1 \leq 0.$$

10. Factor:

$$(n-1)^2 \leq 0.$$

For $n > 1$, $(n-1)$ is strictly positive, and its square is positive. This contradicts the inequality. Therefore, the initial assumption must be false, proving the lemma. □

Corollary 3 (*Consecutive Integer Product Constraint*)**.** *Let $n > 1$ be an integer. Suppose its square root $\sqrt{n}$ does **not** satisfy*

$$\big(\lfloor\sqrt{n}\rfloor + 0.5\big) \;-\; \sqrt{n} \;<\; \frac{1}{\lfloor\sqrt{n}\rfloor}$$

Then n cannot be written as the product of two consecutive positive integers.

Proof. Assume, for contradiction, that $n = c \times (c-1)$ for some integer $c > 1$, yet

$$\big(\lfloor\sqrt{n}\rfloor + 0.5\big) \;-\; \sqrt{n} \;\geq\; \frac{1}{\lfloor\sqrt{n}\rfloor}$$

Note that $c - 1 < \sqrt{n} < c$. The arithmetic mean of c and $(c-1)$ is

$$\frac{c+(c-1)}{2} \;=\; c \;-\; \tfrac{1}{2},$$

while the geometric mean is $\sqrt{c(c-1)}$. By the Square Root Limitation Lemma, whenever two consecutive integers produce n, the difference between this arithmetic mean and the geometric mean $\sqrt{n}$ must be **strictly** less than $\frac{1}{c-1}$.

Next, observe that $\lfloor\sqrt{n}\rfloor = c - 1$. Translating the lemma's requirement:

$$\big(\lfloor\sqrt{n}\rfloor + 0.5\big) \;-\; \sqrt{n} \;=\; \frac{2c-1}{2} \;-\; \sqrt{c(c-1)} \;<\; \frac{1}{c-1}$$

Hence if

$$\big(\lfloor\sqrt{n}\rfloor + 0.5\big) \; - \; \sqrt{n} \; \geq \; \frac{1}{\lfloor\sqrt{n}\rfloor},$$

this inequality cannot hold, contradicting $n = c(c-1)$.

Therefore, if the above bound fails, no integer c can satisfy $n = c(c-1)$. Equivalently, if

$$\big(\lfloor\sqrt{n}\rfloor + 0.5\big) \; - \; \sqrt{n} \; \not< \; \frac{1}{\lfloor\sqrt{n}\rfloor}$$

then n is not the product of two consecutive integers. This completes the proof. □

Proof. **4. Main Proof**

4.1. Transforming the ABC Equation

Starting with the given ABC Factorial Equation:

$$a! \times b! = c!. \tag{1}$$

Dividing by $b!$, it can be rearranged as:

$$a! = \frac{c!}{b!}. \tag{2}$$

Canceling factors from the ends of $c!$ and $b!$, implies that $a!$ equals the product of integers from c down to $b+1$ as justified by Lemma 15:

$$a! = c \times (c-1) \times \cdots \times (b+1). \tag{3}$$

Let $k = c - b$. Then, substituting $c - k$ for b:

$$a! = P_k(c) = c \times (c-1) \times \cdots \times (c-k+1). \tag{4}$$

This rearranges the question into finding any k-tuple products, $P_k(c)$ that are equal to some $a!$ with $1 < a < b < c-1$. The proof begins with case $k = 2$ because for $k = 1$ the equation collapses to the trivial case $a! = c$, which is true for all c, where c is a factorial number.

For case $k = 2$ the question is to determine if there exists an $a!$ such that:

$$a < b \text{ and } b = c-2 \text{ and } a! = c \times (c-1).$$

By Lemma 12 (Larger Factorial Prime) all c can be eliminated where c is a prime number, because if $c \times (c-1)$ is a factorial, and c is prime, then $c \times (c-1)$ has a larger prime factor than $b!$, and then $a > b$, which is a contradiction.

By the same reasoning, all c can be eliminated where c equals a prime number plus one because then $c - 1$ would be that prime number, and $c \times (c-1)$ has the same problem as if c were the prime.

This same problem will arise in all the cases where $k \geq 2$, but c is not at least k greater than the largest prime number less than c. This is because, again, no $a!$ can be less than $(c-k)! = b!$ if the largest prime from $c!$ falls into the quotient, $\frac{c!}{b!}$ which is a candidate for $a!$, as established by Lemma 17 (Limiting k by $c - p_c$). Another way to say this is that the value of c must be in a "gap" between consecutive prime numbers, and at least k above the lower prime.

For example, suppose $c = 16$ and $b = 14$, then $k = 2$ and the candidate for $a!$ would be:

$$\frac{16!}{14!} = 16 \times 15 = 240 = 2^4 \times 3 \times 5$$

which is π-complete, but is not a factorial. However, if the algorithm kept testing to $b = 12$, then $k = 4$ and the candidate for $a!$ would be:

$$\frac{16!}{12!} = 16 \times 15 \ \times 14 \times 13 = 43{,}680 = 2^5 \times 3 \times 5 \times 7 \times 13$$

which fails because it is missing the prime, 11, and because having picked up the largest prime factor of $c!$, 13, the candidate for $a!$ now has a prime factor larger than any in $b!$ so even if it were a factorial number, $a!$ would be greater than $b!$ by lemma 14, which is a contradiction.

This limitation on the maximum value of k by the size of the prime gap at c is a central constraint in this proof because no k-tuple product need be considered as a candidate for $a!$ if it is longer than the prime gap around c is wide.

4.2. Understanding the Product $P_k(c)$

The product $P_k(c)$ involves k consecutive integers descending from c to $c - k + 1$. The goal is to analyze the exponents in the prime factorization of k-tuple products $P_k(c)$ and compare them to those in any possible $a!$ as c becomes larger.

For $a! = P_k(c)$ to hold, Lemma 7 implies that $P_k(c)$ must have the π-complete property. This divides into two classes of cases, those where $1 < k \leq 5$, and the other cases where $k > 5$.

4.2.1. $a! \neq P_k(c)$ for $c > 633,556$ and $1 < k \leq 5$

Looking again at the case $k = 2$, the question is now if $c \times (c-1)$ is π-complete? If not, then $P_2(c)$ is not a factorial number. What are the possible values of c where it could be? A computer search program, π-complete_Analyzer, listed in Listing 2 of Appendix F, has been written and run for the purpose of finding these possible values. In the case $k = 2$ it locates these c values for π-complete products, $P_2(c)$:

$c \in$ {2, 3, 4, 6, 9, 10, 15, 16, 21, 25, 36, 81, 126, 225, 385, 441, 540, 715, 1,716, 2,080, 2,401, 3,025, 4,375, 9,801, 12,376, 123,201, 194,481, 633,556}

Table 1 shows this set above $c = 9$ and the prime factorizations of the resulting products, $P_2(c)$. (See Appendix F for a detailed discussion of the operation of π-complete_Analyzer, and the extent of its search space.)

Of these values for $c > 9$ having a π-complete $P_2(c)$, only a few ending in $c =$ 2,080, give a $P_2(c)$ with the FEF property, and of those, none are factorial numbers.

4.2.2. No π-complete $P_2(c)$ for $c >$ 633,556

However, how can it be shown that there are no π-complete $P_2(c)$ for $c >$ 633,556? The theory for this conclusion comes from both the rare nature of π-complete numbers and the squeezing of the square root of products of consecutive integers against their arithmetic mean as those consecutive integers get larger. First, although the set of all π-complete numbers is without bound, the numbers appear less often moving up the number line. Whereas there are 18 π-complete numbers from 2 to 100, there are only 254 π-complete numbers from 101 to 100,000. Going from 100,001 to 1,000,000 only adds another 434. While these get harder to find, what is even harder to find is a π-complete product of two consecutive large integers. This is because for any large consecutive integers, the square root of their product (geometric mean) is squeezed closer and closer to their arithmetic mean.

Consider 2×1. Here, 2 is 100% greater than 1. Their product is 2 and its square root is 1.4142, while the arithmetic mean of the pair is 1.5. Moving on to 10×9, 10 is only 11% larger than 9; the square root of their product is 9.4868 and their arithmetic mean is 9.5. Whereas 2×1 had 1.5 - 1.4142 = 0.0858 room to get that square root into, 10×9 has only 9.5 - 9.4868 = 0.0132. Moving up to c = 2,080, the room left is only

$$\frac{(2{,}080 + 2{,}079)}{2} - \sqrt{2{,}080 \times 2{,}079} = 0.00006011$$

As the value of a π-complete number, $r\#m$, gets very large, it becomes more and more rare for the prime factorization of that number to be able to be partitioned into the product of two consecutive integers $c \times (c-1)$ (where c and $c-1$ share no prime factors as consecutive integers are coprime). Whereas the products of consecutive integers go on without limit, the subset of those that are π-complete reaches a limit because the requirement on $r\#m$ of having all prime factors from 2, up to and including p_r with positive integer exponents, becomes in conflict with the requirement that

$$\lfloor\sqrt{r\#m}\rfloor + 0.5 - \sqrt{r\#m} < \frac{1}{\lfloor\sqrt{r\#m}\rfloor}$$

as required by Corollary 3 of Lemma 18 (Square Root Limitation).

Thus, it is shown by theory that the sequence of π-complete products of two consecutive integers must terminate for large c, and empirical search has verified this, finding the last value possible for c to be 633,556. (See Appendix A for a proof by induction of this for each π-complete Order, $\mathcal{P}(r)$.)

An additional argument can be made by considering the role of m in the π-complete representation $r\#m$. For a π-complete number to be expressed as the product of consecutive integers, its square root must fall between those integers. This requires m to balance the irrational contribution of $\sqrt{r\#}$, as seen in the case of $633{,}556 \times 633{,}555$, where m = 41,382. Here, m balances the contribution of 19 (the largest prime in $8\#$) and provides additional factors, such as 3^2 and 11^2, which are perfect squares, and 2, which returns the square root to the vicinity of the half integer as in $\sqrt{7\#}$.

However, when moving to $9\#$, the introduction of a new prime (23) fundamentally disrupts this balance. Any m that could potentially restore balance would need to incorporate 23 into its factorization. This increases both the magnitude of $r\#m$ and the precision required for its square root to land between consecutive integers. The resulting product becomes too large for its square root to fit within the narrowing window defined by the arithmetic mean and geometric mean gap. Thus, no m can provide such a balance for 9# or higher primorials, ensuring that no two consecutive integer π-complete products exist beyond $633{,}556 \times 633{,}555$.

4.2.3. Case $k = 3$

In case $k = 3$ the same kind of quantization limitations apply, except now the product $P_3(c)$ grows with the cube of c, and the prime factorizations of $c, (c - 1)$ and $(c - 2)$ must be combined to make a π-complete product. As consecutive integers, c and $(c - 1)$ are coprime, and $(c - 1)$ and $(c - 2)$ are coprime, and although c and $(c - 2)$ share the prime 2, they are coprime for all the rest of their prime factorization. This is more restrictive than the $k = 2$ case, and the set of π-complete values allowed for c becomes: {3, 4, 5, 6, 7, 10, 16, 22, 50, 56, 100, 352, 442} Both $P_3(3)$ and $P_3(4)$ are factorial numbers, but they fall into the cases were a could not be greater than 1 and still less than b.

This sequence terminates quickly at 442, but there is a solution at 10 because $10 \times 9 \times 8 = 720 = 6!$ which is the known solution $(a, b, c) = (6, 7, 10)$. The $P_3(c)$ of all the other numbers in this set, are not solutions.

4.2.4. Case $k = 4$

For the π-complete $P_4(c)$, the allowed c value set is: {4, 5, 6, 7, 8, 10, 66} Both $P_4(4)$ and $P_4(5)$ are factorial numbers, but again they fall into the cases were a could not be greater than 1 and still less than b, and the rest are not factorials.

4.2.5. Case $k = 5$

For $P_5(c) = c \times (c - 1) \times \cdots \times (c - 4)$ the π-complete values occur where c is an element of the set: {5, 6, 7, 8, 10, 11, 14, 15}. Both $P_5(5)$ and $P_5(6)$ are factorial numbers, but again they fall into the cases were a could not be greater than 1 and still less than b, and the rest are not factorials.

Thus, it has been shown that for all values of c and all values of $1 < k \leq 5$ the product $P_k(c)$ either is not π-complete so cannot be a factorial, or is a very small factorial but cannot be a solution where $1 < a < b < c - 1$, or it is the solution at $P_3(10)$.

This data is summarized for values of $c > 9$ in Table 1 below. Because all of the values shown are π-complete, the prime factorizations for the multipliers, m, for each value are simply the prime factorizations shown with 1 subtracted from each exponent to divide away the primorial base. For example, in the last entry the exponents are: 2, 3, 1, 1, 3, 1, 1, 2 so subtracting the primorial exponents gives $m = 2 \times 3^2 \times 11^2 \times 19 = 41{,}382$.

Product	**Prime Factorization**	**Properties**
$P_2(10)$	$2^1 \times 3^2 \times 5^1$	
$P_3(10)$	$2^4 \times 3^2 \times 5^1$	(FEF) (6!)
$P_2(15)$	$2^1 \times 3^1 \times 5^1 \times 7^1$	(4#)
$P_5(15)$	$2^3 \times 3^2 \times 5^1 \times 7^1 \times 11^1 \times 13^1$	(FEF)
$P_2(16)$	$2^4 \times 3^1 \times 5^1$	(FEF)
$P_2(21)$	$2^2 \times 3^1 \times 5^1 \times 7^1$	(FEF)
$P_3(22)$	$2^3 \times 3^1 \times 5^1 \times 7^1 \times 11^1$	(FEF)
$P_2(25)$	$2^3 \times 3^1 \times 5^2$	
$P_2(36)$	$2^2 \times 3^2 \times 5^1 \times 7^1$	(FEF)
$P_3(56)$	$2^4 \times 3^3 \times 5^1 \times 7^1 \times 11^1$	(FEF)
$P_4(66)$	$2^7 \times 3^3 \times 5^1 \times 7^1 \times 11^1 \times 13^1$	(FEF)
$P_2(81)$	$2^4 \times 3^4 \times 5^1$	(FEF)
$P_3(100)$	$2^3 \times 3^2 \times 5^2 \times 7^2 \times 11^1$	(FEF)
$P_2(126)$	$2^1 \times 3^2 \times 5^3 \times 7^1$	
$P_2(225)$	$2^5 \times 3^2 \times 5^2 \times 7^1$	(FEF)
$P_3(352)$	$2^6 \times 3^3 \times 5^2 \times 7^1 \times 11^1 \times 13^1$	(FEF)
$P_2(385)$	$2^7 \times 3^1 \times 5^1 \times 7^1 \times 11^1$	(FEF)
$P_2(441)$	$2^3 \times 3^2 \times 5^1 \times 7^2 \times 11^1$	
$P_3(442)$	$2^4 \times 3^2 \times 5^1 \times 7^2 \times 11^1 \times 13^1 \times 17^1$	
$P_2(540)$	$2^2 \times 3^3 \times 5^1 \times 7^2 \times 11^1$	
$P_2(715)$	$2^1 \times 3^1 \times 5^1 \times 7^1 \times 11^1 \times 13^1 \times 17^1$	(7#)
$P_2(1{,}716)$	$2^2 \times 3^1 \times 5^1 \times 7^3 \times 11^1 \times 13^1$	
$P_2(2{,}080)$	$2^5 \times 3^3 \times 5^1 \times 7^1 \times 11^1 \times 13^1$	(FEF)
$P_2(2{,}401)$	$2^5 \times 3^1 \times 5^2 \times 7^4$	
$P_2(3{,}025)$	$2^4 \times 3^3 \times 5^2 \times 7^1 \times 11^2$	
$P_2(4{,}375)$	$2^1 \times 3^7 \times 5^4 \times 7^1$	
$P_2(9{,}801)$	$2^3 \times 3^4 \times 5^2 \times 7^2 \times 11^2$	
$P_2(12{,}376)$	$2^3 \times 3^2 \times 5^3 \times 7^1 \times 11^1 \times 13^1 \times 17^1$	
$P_2(194{,}481)$	$2^4 \times 3^4 \times 5^1 \times 7^4 \times 11^1 \times 13^1 \times 17^1$	
$P_2(633{,}556)$	$2^2 \times 3^3 \times 5^1 \times 7^1 \times 11^3 \times 13^1 \times 17^1 \times 19^2$	

Table 1: Prime Factorizations of the π-complete Values of $P_k(c)$

4.2.6. Proof for cases $k > 5$

Unlike the cases $k < 6$ above, where P_{cm} decreases for each successive value of k, the multiple factors in the tuple begin to provide more room for primes at $k = 6$. For example, with small values of c, such as k = 6 and c = 14,

$$P_6(14) = 14 \times 13 \times 12 \times 11 \times 10 \times 9$$

is π-complete, while:

$$P_6(25) = 25 \times 24 \times 23 \times 22 \times 21 \times 20$$

is missing primes 19, 17, and 13, so it is not. It is characteristic for these long k-tuples to be missing a string of primes just below their lowest multiplier.

The increasing c value limit, P_{cm}, on π-complete k-tuples, $P_k(c)$, starting at c, has been studied by empirical computation. Table 2 below shows the computed maximum values for c that will result in $P_k(c)$ less than P_{cm} for k values from 6 to 495. If the value of c is above these limits for k in the listed range, then the k-tuple is not π-complete, and therefore, does not have the possibility of being a factorial number, for any $a!$.

k range	**max(c)**	**k range**	**max(c)**
6-9	19	223-240	495
10-15	31	241-246	507
18-25	53	247-252	519
26-33	69	253-260	533
34-39	79	261-268	549
42-46	99	269-274	571
47-54	113	275-298	609
55-62	129	299-320	655
63-66	137	323-340	691
67-74	153	343-346	709
77-80	169	347-356	727
81-92	189	364-370	753
93-98	199	373-376	769
99-122	249	377-388	789
125-128	265	389-394	807
129-144	293	395-406	829
145-150	307	407-422	857
151-158	321	423-430	873
161-166	339	433-436	889
167-170	349	437-448	909
171-186	377	451-466	949
187-198	409	467-478	969
199-214	441	479-490	993
215-219	451	493-495	1020

Table 2: Maximum Values of c Where $P_k(c)$ is π-complete for Ranges of k

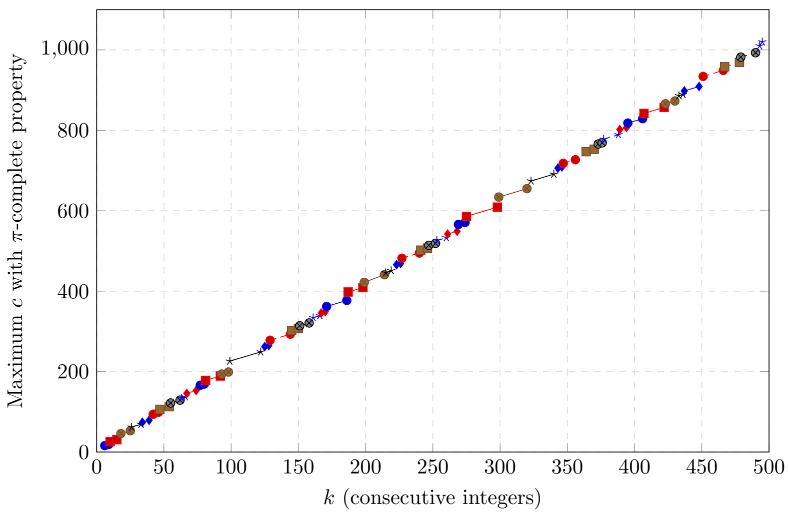

Figure 1: π-complete maximum values for k-tuple products shown as line segments connecting sequential groups

As c grows and enters larger prime gaps, where k can get larger, the limit of P_{cm} increases with increasing prime gap size. It is well known [2] that the prime gap size grows on the order of ln c for large c. An approximation for the bound on c that provides a possible factorial number for $P_k(c)$, as shown in Table 2, above, for values of $k > 5$, can be given by $2 \times k$, as plotted in Figure 1. When this is combined with the prime gap growth restriction on k, the growth of this bound on c for $k > 5$ is on the order of $2 \ln c$.

The size of the prime gap at the prime 1,349,533 is 118, so $c = 633{,}556$ is at a gap size of less than 118, but that gives a large k bound on c of less than $c = 129$. As shown in Table 2, the rate of the growth of the large k bound on c will never catch the linear growth of c, even though prime gaps grow without limit. This is in accord with the limit on $c - b$ reported by Hajdu et al.[5]:

$$c - b \leq 5 \ln \ln c$$

for large c, indicating that while c continues to increase linearly, the boundary where $P_k(c) > P_{\mathrm{cm}}$ will become even farther below c, eliminating any possible factorial equality to $P_k(c)$ no matter how large the prime gap at c becomes.

4.3. $a! \neq P_k(c)$ for all $10 < c < 633{,}556$

As it has been shown that no $P_k(c)$ can have the π-complete property, given all possible k at all values of $c > 633{,}556$, there are no possible factorials, $a!$, for these values of c. It remains to check for solutions to the ABC Factorial Equation at

$c \leq 633{,}556$, which has been accomplished in the literature[4], and by the computer software in Appendix E, for all values of $c \leq 10{,}000{,}000$. This has found only the single known solution.

Thus, it is demonstrated that the only non-trivial solution to the ABC Factorial Equation is:

$$(a, b, c) = (6, 7, 10)$$

therefore, the ABC Factorial Conjecture is true. □

Corollary 4 (Primorial Doublets)**.** *The complete set of pairs of consecutive positive integers, whose product is a primorial number is:*
$\{$*(1, 2), (2, 3), (5, 6), (14, 15), (714, 715)*$\}$.

Those products are:

$$\begin{aligned} 1 \times 2 &= 2 \text{ which is } 1\#, \\ 2 \times 3 &= 6 \text{ which is } 2\#, \\ 5 \times 6 &= 30 \text{ which is } 3\#, \\ 14 \times 15 &= 210 \text{ which is } 4\#, \\ 714 \times 715 &= 510{,}510 \text{ which is } 7\#. \end{aligned}$$

No other possibilities exist because all primorials, $r\#$, are required by definition to have a complete set of prime factors up to prime number r. Therefore all primorials are required to have the π-complete property. However it has been shown that above (633,555, 633,556), no consecutive integers can produce a product that is π-complete, and all π-complete products below this limit have been examined and found not to be primorial numbers, except for those listed above.

The impossibility of finding consecutive integer products equal to primorials beyond 7# can be understood through the interaction of two fundamental constraints. First, for any primorial $r\#$, its square root must be extremely close to a half-integer for it to be expressible as a product of consecutive integers. Second, each new prime factor introduced creates an irrational component in the square root that pushes it further from any half-integer value. For $8\# = 7\# \times 19$, the introduction of 19 forces the square root to deviate from any half-integer by more than $\frac{1}{c}$ where $c \approx \sqrt{8\#}$, making it impossible to satisfy both the π-complete property and the consecutive integer requirement. This effect becomes more pronounced with each subsequent prime factor, ensuring that no primorial beyond 7# can be expressed as a product of consecutive integers.

If primorials are factored into integer pairs, (d, e), that bracket the square root of the primorial value with $d < e$, then the pairs with the minimal numeric difference, $\mathcal{F}_\delta(n) = (e - d)$, are shown in Table 3, with their differences, for primorials 4# through 19#.

r	**Factor** d	**Factor** e	**Sqrt(#)**	$\mathcal{F}_\delta(r\#)$
4	14	15	14.49138	1
5	42	55	48.06246	13
6	165	182	173.29166	17
7	714	715	714.49983	1
8	3,094	3,135	3,114.43253	41
9	14,858	15,015	14,936.29372	157
10	79,534	81,345	80,434.40327	1,811
11	447,051	448,630	447,839.80409	1,579
12	2,714,690	2,733,549	2,724,103.17991	18,859
13	17,395,070	17,490,603	17,442,771.09657	95,533
14	114,371,070	114,388,729	11,4379,899.15921	17,659
15	783,152,070	785,147,363	784,149,081.86422	1,995,293
16	5,708,587,335	5,708,795,638	5,708,691,485.54991	208,303
17	43,848,093,003	43,850,489,690	43,849,291,330.12542	2,396,687
18	342,444,658,094	342,503,171,205	342,473,913,399.84856	58,513,111
19	2,803,119,896,185	2,803,419,704,514	2,803,269,796,341.45580	299,808,329
20	23,619,540,863,730	23,622,001,517,543	23,620,771,158,594.69481	2,460,653,813
21	201,813,981,102,615	201,817,933,409,378	201,815,957,246,321.39244	3,952,306,763
22	1,793,779,293,633,437	1,793,779,635,410,490	1,793,779,464,521,955.35996	341,777,053
23	16,342,050,964,565,644	16,342,166,369,958,702	16,342,108,667,160,301.66527	115,405,393,058

Table 3: Search for Minimum Primorial Factor Delta

Whereas this table shows consecutive integer factors at 4# and 7#, the difference between factors builds to 17 at 6#, then drops and rebuilds to 1,811 at 10#, drops to 1,579 at 11#, rebuilds to 95,533 at 13#, drops again to 17,659 at 14#, goes up again to 1,995,293 at 15#, drops back to 208,303 at 16# and finally builds to 299,808,329 at 19#. This "uneven stair climbing" behavior represents increasing intervals punctuated by opportunities where the gaps between successive prime numbers give a factor packing fit closer to the square root. Those "falling back" differences to closer fits make their own sequence of $\mathcal{F}_\delta$ values: 1, 1,579, 17,659, 208,302, 341,777,053 which keep getting larger.

If a primorial number were the product of two consecutive candidate integers, d and e, then:

$$d = \lfloor \sqrt{r\#} \rfloor \text{ and } e = \lceil \sqrt{r\#} \rceil \text{ so } \mathcal{F}_\delta(r\#) = 1.$$

Table 4 shows the primorials 4# through 23# split into candidate factors d and e. As shown, this provides the correct factorization for 4# and 7#, but after 7# all candidates d and e have large prime factors that do not belong in a primorial. Unable to squeeze the square root close enough to the arithmetic mean, this will continue as the reciprocal of the square root approaches zero.

In the Python package, mpmath, it is possible to set the significant number of decimal digits of floating point calculations to 100. Doing so and taking the square roots of the primorials up to 95# shows that none can get the fractional part of the square root to even within 0.001 (that is, 0.5 - 0.499) of the arithmetic mean of an integer pair straddling that square root. The closest in that group is 65# which has a square root of:

7,815,173,028,534,505,588,363,874,620,346,130,829,799,138,281,055,400,452,651,446,121.490817472843715547232
and the reciprocal of that is smaller than 10^{-63} so it is squeezed out.

r	**Candidate** d	**Factorization** d	**Candidate** e	**Factorization** e
4	14	2×7	15	3×5
5	48	$2^4 \times 3$	49	7^2
6	173	173	174	$2 \times 3 \times 29$
7	714	$2 \times 3 \times 7 \times 17$	715	$5 \times 11 \times 13$
8	3,114	$2 \times 3^2 \times 173$	3,115	$5 \times 7 \times 89$
9	14,936	$2^3 \times 1867$	14,937	$3 \times 13 \times 383$
10	80,434	$2 \times 131 \times 307$	80,435	5×16087
11	447,839	7×63977	447,840	$2^5 \times 3^2 \times 5 \times 311$
12	2,724,103	251×10853	2,724,104	$2^3 \times 167 \times 2039$
13	17,442,771	3×5814257	17,442,772	$2^2 \times 1049 \times 4157$
14	114,379,899	$3 \times 131 \times 291043$	114,379,900	$2^2 \times 5^2 \times 1143799$
15	784,149,081	$3 \times 47 \times 5561341$	784,149,082	$2 \times 367 \times 1068323$
16	5,708,691,485	$5 \times 7 \times 433 \times 376687$	5,708,691,486	$2 \times 3^3 \times 105716509$
17	43,849,291,330	$2 \times 5 \times 13 \times 3677 \times 91733$	43,849,291,331	3347×13101073
18	342,473,913,399	3×114157971133	342,473,913,400	$2^3 \times 5^2 \times 59 \times 563 \times 51551$
19	2,803,269,796,341	$3 \times 7 \times 133489037921$	2,803,269,796,342	2×1401634898171
20	23,620,771,158,594	$2 \times 3 \times 3936795193099$	23,620,771,158,595	$5 \times 13 \times 1013 \times 25747 \times 13933$
21	201,815,957,246,321	$11 \times 67 \times 273834406033$	201,815,957,246,322	$2 \times 3 \times 41 \times 1327 \times 618229141$
22	1,793,779,464,521,955	$3 \times 5 \times 119585297634797$	1,793,779,464,521,956	$2^2 \times 79 \times 5676517292791$
23	16,342,108,667,160,302	$2 \times 13 \times 628542641044627$	16,342,108,667,160,303	$3 \times 17 \times 67 \times 4782589601159$

Table 4: Candidates d, e, for Primorial Consecutive Factors

5. Application of π-complete Numbers to Brocard's Equation

Brocard's equation, expressed as $m! + 1 = n^2$, was first posed by Henri Brocard (1845-1922), a French mathematician and meteorologist. Brocard introduced this problem in two notes published in the Bulletin de la Société Mathématique de France in 1876 and 1885. While primarily known for his work in geometry (particularly the Brocard points and Brocard circle in triangle geometry), his number-theoretic inquiries have proven equally enduring.

During his initial investigations, Brocard identified two solutions to the equation:

$$(m, n) = (4, 5), \text{ as } 4! + 1 = 24 + 1 = 25 = 5^2$$

and

$$(m, n) = (5, 11), \text{ as } 5! + 1 = 120 + 1 = 121 = 11^2.$$

The third solution, $(m, n) = (7, 71)$, was discovered later by Ramanujan in the early 20th century. This demonstrated that

$$7! + 1 = 5040 + 1 = 5041 = 71^2.$$

That discovery appeared in his notebooks but was not formally published during his lifetime.

For many years, these three solutions were the only ones known, but the question of whether additional solutions existed remained open. Paul Erdős, who had a particular interest in Diophantine equations, conjectured that no further solutions existed, but a formal proof remained elusive. In this proof, it will be demonstrated that these are the only solutions by applying techniques from the prior analysis of the ABC Factorial Equation, particularly the concepts of π-complete numbers and the Factorial Exponent Form property.

Proof. The equation $m! + 1 = n^2$ can be rewritten as:

$$(n-1)(n+1) = m!.$$

This form reveals Brocard's Problem as a question about when the product of two consecutive even integers can equal a factorial, aligning perfectly with prior techniques for analyzing prime factorization patterns in such products. It has been shown that any number that is not π-complete cannot be a factorial, therefore if it can be shown, as in the case of $c(c-1)$, that the product $(n+1)(n-1)$ lacks the π-complete property for all n larger than the known solutions, then the conjecture that there are no other solustions is proven.

A slight variation of the Pi_Analyzer.py was used to check the π-completeness of the product $(n+1)(n-1)$ for n up to 100,000,000. The results are shown in the following list:

$2 \times 4 = 2^3$.
$4 \times 6 = 2^3 \times 3$ FEF $=$ $4!$.
$6 \times 8 = 2^4 \times 3$ FEF.
$10 \times 12 = 2^3 \times 3 \times 5$ FEF $=$ $5!$.
$16 \times 18 = 2^5 \times 3^2$.
$18 \times 20 = 2^3 \times 3^2 \times 5$ FEF.
$28 \times 30 = 2^3 \times 3 \times 5 \times 7$ FEF.
$30 \times 32 = 2^6 \times 3 \times 5$ FEF.
$40 \times 42 = 2^4 \times 3 \times 5 \times 7$ FEF.
$48 \times 50 = 2^5 \times 3 \times 5^2$.
$70 \times 72 = 2^4 \times 3^2 \times 5 \times 7$ FEF $=$ $7!$.
$160 \times 162 = 2^6 \times 3^4 \times 5$ FEF.
$250 \times 252 = 2^3 \times 3^2 \times 5^3 \times 7$.
$448 \times 450 = 2^7 \times 3^2 \times 5^2 \times 7$ FEF.
$768 \times 770 = 2^9 \times 3 \times 5 \times 7 \times 11$ FEF.
$880 \times 882 = 2^5 \times 3^2 \times 5 \times 7^2 \times 11$.
$1{,}078 \times 1{,}080 = 2^4 \times 3^3 \times 5 \times 7^2 \times 11$.
$1{,}428 \times 1{,}430 = 2^3 \times 3 \times 5 \times 7 \times 11 \times 13 \times 17$ FEF.
$3{,}430 \times 3{,}432 = 2^4 \times 3 \times 5 \times 7^3 \times 11 \times 13$.

$4{,}158 \times 4{,}160 = 2^7 \times 3^3 \times 5 \times 7 \times 11 \times 13$ *FEF*.
$4{,}800 \times 4{,}802 = 2^7 \times 3 \times 5^2 \times 7^4$.
$6{,}048 \times 6{,}050 = 2^6 \times 3^3 \times 5^2 \times 7 \times 11^2$.
$8{,}748 \times 8{,}750 = 2^3 \times 3^7 \times 5^4 \times 7$.
$19{,}600 \times 19{,}602 = 2^5 \times 3^4 \times 5^2 \times 7^2 \times 11^2$.
$24{,}750 \times 24{,}752 = 2^5 \times 3^2 \times 5^3 \times 7 \times 11 \times 13 \times 17$.
$246{,}400 \times 246{,}402 = 2^8 \times 3^6 \times 5^2 \times 7 \times 11 \times 13^2$.
$388{,}960 \times 388{,}962 = 2^6 \times 3^4 \times 5 \times 7^4 \times 11 \times 13 \times 17$.
$1{,}267{,}110 \times 1{,}267{,}112 = 2^4 \times 3^3 \times 5 \times 7 \times 11^3 \times 13 \times 17 \times 19^2$.

As was the case for $c(c-1)$, there are factorials such as, 4!, 5! and 7! in the lower primorial orders, but products stop having the FEF property in the higher products, and then there is only one last π-complete product in order 8. A close look at these products reveils the same set of π-complete factorizations, except four times larger (the exponent of 2 is 2 larger). This is reasonable because the set of pairs of consecutive even integers is the same as the set of consecutive integers where each has been doubled, yeilding a product that is four times larger. Algebraicaly if:

$$n = 2c - 1$$

then:

$$(n+1)(n-1) = (2c-1+1)(2c-1-1) = 4c(c-1).$$

This means that the product values are four times those which are proven bounded above and in Appendix A. If a number is π-complete, then multiplying by 4 just adds 2 to the first prime exponent (of 2), but the number remains π-complete. All these products are even numbers, so if any even product is not π-complete, multiplying by 4 is not going to make it π-complete because it already has 2 as a prime factor. Thus, given that $633{,}556 \times 633{,}555$ was the last π-complete product of consecutive integers, $1{,}267{,}110 \times 1{,}267{,}112$ has to be the last π-complete product of consecutive even integers because if there were a larger value, that value divided by 4 would be in list from $c(c-1)$.

Again, as was shown for $c(c-1)$, the prime factorization of the product $(n+1)(n-1)$ cannot have the addition of any prime factor past 19 and still remain π-complete. Thus the conjecture by Paul Erdős is proved. □

6. Final Remarks

This proof extends the work of Hajdu et al.[5], who showed that for $b - a \leq 10^6$, the equation $a! \times b! = c!$ has only one non-trivial solution $(a, b, c) = (6, 7, 10)$. Their approach relied on explicit bounds for the prime counting function $\pi(x)$ and analysis of prime-free intervals.

The method presented here demonstrates that no solutions exist beyond $c =$ 633,556 through a novel analysis of prime factor distribution in consecutive integer products. This provides a complete characterization of the solution space by:

1. Establishing that $c = 633{,}556$ represents the largest possible value in any k-tuple of consecutive integers having the π-complete property.

2. Proving that for any $c > 633{,}556$, the prime factorization pattern necessary for $a! \times b! = c!$ cannot occur.

This extends the limitations found by Habsieger[4] and Hajdu et al. by showing that no solutions can exist beyond those bounds, regardless of the size of c. The result follows from analysis of prime gaps and the distribution of prime factors in products of consecutive integers, providing a different perspective on the structural constraints that force $(6, 7, 10)$ to be unique.

Furthermore, the application of π-complete number properties to Brocard's equation demonstrates the broader applicability of this approach to other Diophantine equations involving factorials. Based on the techniques developed in this work, the author proposes the following conjecture:

For any polynomial $P(x)$ of degree 2 or higher with integer coefficients, there exists a finite bound B such that for all $x > B$, the value $P(x)$ is not π-complete. Consequently, for any equation of the form $n! = P(x)$ where n and x are positive integers, there exist only finitely many solutions.

While a comprehensive proof of this conjecture is beyond the scope of the current work, the methodology established here provides a foundation for approaching this broader class of problems.

While the ABC Factorial equation was restricted by $1 < a < b < c-1$, a need not be restricted from equaling b because if $a = b$, then the equation becomes $(a!)^2 = c!$ which cannot be true because from Corollary 1 no factorial can be a power, greater than 1, of any other number because its top prime exponent must be 1, and would be a multiple of that power, otherwise.[3]

The numeric research program, Factorial_Search.py (FS), is shown in listing 1 of Appendix E. It is through the development of this program that the number theory discoveries presented in this proof were made. FS searches up to a given limit for values of c that satisfy the equation $a! \times b! = c!$ for b that is greater than a and less than $c - 1$. To do this, FS subtracts the prime factorization exponents of $b!$ from the corresponding exponents of $c!$ and checks the result against the prime factorizations of $a!$ starting at $a = 3$ and going up from there. It quickly finds that when $c = 10$ and $b = 7$, $a!$ will be a match at $a = 6$. It should be noted that this program works on additions and subtractions of integer exponents of prime factorizations without any resort to floating point numbers and is therefore free of numeric resolution issues.

While FS searches the values of c, it keeps statistics about the range of b values it tries and the number of attempts it makes to match an $a!$. In order to speed the FS search, number theory principles were used to avoid testing cases that theory ruled out. For example, the length of the prime factorization for $b!$ must be the same as the length for $c!$ or the subtraction would drop the largest prime factor in $c!$ into the quotient, and no $a!$ could have that because $a < b$. Also, if the top prime factor exponent of the quotient is not exactly one, the quotient cannot be a factorial. The same is true if the exponents of the quotient do not proceed from the exponent of 2 onward without increasing. This implies that no exponent in the quotient factorization can be zero from 2 to the top prime factor. Adding these restrictions greatly improved the speed of the FS program.

From the statistics generated by FS, a curious behavior was noticed. FS would stop trying to match any quotient of $c!/b!$ for all values of $c >$ 2,080. FS has been run up to $c = 10,000,000$ but still no attempts to match the quotient after $c =$ 2,080 have been found. These tests required the quotient to have the FEF property defined above in this proof, and no quotient was found to have the FEF property past the special product 2,080 $\times$ 2,079.

As an experiment, some restrictions were removed from FS and runs were repeated. With only the π-complete requirement imposed upon the quotient, FS would stop trying to find any factorial match past $c =$ 633,556. This led to testing the properties of the prime factorizations of products of consecutive integers, in general, which led to the discovery that for k-tuples of $1 < k < 6$ the π-complete limit dropped rapidly from $c =$ 633,556 to $c = 15$, and that the π-complete limit grew slowly for $k > 5$. These observations led directly to this proof.

Even if there were more π-complete products of consecutive integers, there is a pattern in the prime factorizations of the terminal k-tuples that shows that extension of these limits will still not allow a k-tuple product to match any factorial number. Table 5 shows the terminal products of the k-tuples listed as their values fit into the progression of the values of factorial numbers from 9! to 15!. Whereas the factorials are building up high exponents for the primes 2 and 3, the k-tuples are not not doing so, but rather, extending their largest prime factor beyond the factorials. Remembering that prime factorizations are unique, a match must be at every exponent, so there is no match by somehow balancing more exponent values on one end against more exponent values on the other.

Product	**Value**	**Prime Factorization**
9!	362,880	$2^7 \times 3^4 \times 5^1 \times 7^1$
$P_5(15)$	1,395,360	$2^3 \times 3^2 \times 5^1 \times 7^1 \times 11^1 \times 13^1$
10!	3,628,800	$2^8 \times 3^4 \times 5^2 \times 7^1$
$P_4(66)$	17,297,280	$2^7 \times 3^3 \times 5^1 \times 7^1 \times 11^1 \times 13^1$
11!	39,916,800	$2^8 \times 3^4 \times 5^2 \times 7^1 \times 11^1$
$P_3(442)$	85,765,680	$2^4 \times 3^2 \times 5^1 \times 7^2 \times 11^1 \times 13^1 \times 17^1$
12!	479,001,600	$2^{10} \times 3^5 \times 5^2 \times 7^1 \times 11^1$
13!	6,227,020,800	$^{10} \times 3^5 \times 5^2 \times 7^1 \times 11^1 \times 13^1$
14!	87,178,291,200	$2^{11} \times 3^5 \times 5^2 \times 7^2 \times 11^1 \times 13^1$
$P_2(633,556)$	401,392,571,580	$2^2 \times 3^3 \times 5^1 \times 7^1 \times 11^3 \times 13^1 \times 17^1 \times 19^2$
15!	1,307,674,368,000	$2^{11} \times 3^6 \times 5^3 \times 7^2 \times 11^1 \times 13^1$

Table 5: $P_k(c)$ Limits Among Factorials

Together with the source code for Factorial_Search.py in listing 1 of Appendix E, the source code of the program Pi-complete_Analyzer.py (PA) is given in listing 2 of Appendix F. When run, PA searches products of consecutive integer k-tuples for values that have the π-complete property. The first section searches $1 < k < 6$ for c up to 1,000,000. The next section searches $5 < k < 26$ to show the sequences of π-complete products for c up to 1,000. Then the last section searches $5 < k < 500$, running up to $c =$ 1,200 but only storing and printing the top limit for each k.

From this data it became obvious that with the limit growth rate for k less than the linear growth rate of c, once c was past 633,556 FS would never again attempt to match $c!/b!$ with any $a!$, so no further solutions could be found. As in the FS program, PA uses integer operations on prime factorization exponents and is not subject to floating point numeric resolution issues. The initial part of PA that searches for products of pairs of integers that are π-complete, has been run to a c value of 100,000,000 without finding another π-complete product past $c =$ 633,556, which is in accord with theory from the Square Root Limitation Lemma.

Acknowledgements.
The author wishes to acknowledge the help of the artificial intelligence program, Claude 3.5 Sonnet, by Anthropic, which was used for literature search, and to help generate the computer programs and LaTeX code presented here, as well as the artificial intelligence program, GPT-o1, by OpenAI for help with proof construction and constructive criticism.

References

[1] T. M. Apostol, *Introduction to Analytic Number Theory*, Springer-Verlag, 1976.

[2] H. Cramér, On the order of magnitude of the difference between consecutive prime numbers, *Acta Arithmetica*, **2**, (1936), 23–46.

[3] P. Erdős, J.L. Selfridge: The product of consecutive integers is never a power, *Illinois J. Math.* **19(2)**, (1975) 292–301

[4] L. Habsieger, Explicit bounds for the Diophantine equation A!B! = C!, *The Fibonacci Quarterly*, **57 (1)**, (2019), 21-28.

[5] L. Hajdu, Á. Papp, T. Szakács, On the equation A!B!=C!, *J. Number Theory* **183**, (2018), 267-279.

[6] G. H. Hardy and S. Ramanujan, The Normal Number of Prime Factors of a Number n, *Quarterly Journal of Mathematics*, **48**, (1917), 76–92.

[7] A. M. Legendre, Théorie des nombres. Firmin Didot Frères, *Paris (1830)*

[8] P. Mihailescu, A class number free criterion for Catalan's conjecture, *Journal of Number theory*, **99** (2003).

A. Bounds on π-complete Products of Consecutive Integers

As stated earlier, an integer $N > 1$ is π-complete of order r, that is, $\in \mathcal{P}(r)$, if, and only if, its prime factorization contains every prime number p_i with exponent $e_i > 0$ up to p_r, its largest prime factor. In this presentation such numbers are written as $N = r\#m$ where $r\#$ is the r-th primorial number and m is smooth with respect to p_r.

$$r\#m \ = \ 2^{\,e_1} \times 3^{\,e_2} \times \cdots \times p_r^{e_r}.$$

It will now be shown that for all π-complete products of consecutive integers, c and $c-1$, there is always a limitation on the maximum value of c.

The consideration of π-complete products of two consecutive integers, $P_2(c)$, may proceed based on the order of the primorial comprising its prime factors, starting with order 2, which comprises 2 and 3. For any such product the fact that c and $c-1$ are coprime means that there is a binary partition of the prime factors (with their respective powers) possible, assigning prime factors to one, f_1, or the other, f_0 side. Order 2, thus, has 2^2 partitions, $S_{2,j}$ for $j \in \ [0,3]$, such that:

$$S_{2,0} \Rightarrow f_1 = 1, \ f_0 = 2^{e_1} \times 3^{e_2}.$$
$$S_{2,1} \Rightarrow f_1 = 2^{e_1}, \ f_0 = 3^{e_2}.$$
$$S_{2,2} \Rightarrow f_1 = 3^{e_2}, \ f_0 = 2^{e_1}.$$
$$S_{2,3} \Rightarrow f_1 = 2^{e_1} \times 3^{e_2}, \ f_0 = 1.$$

This binary partitioning arrangement, $S_{r,j}$ assigns prime p_i and its exponent e_i to be a multiplier for either f_1 or f_0, exclusively, based on the $(i-1)$-th binary bit of j, where j goes from 0 to $2^r - 1$. However, the partitions assigning all prime factors to only one side are not considered, leaving 2 symmetric partitions:

$$f_1 = 2^{e_1}, \ f_0 = 3^{e_2} \text{ and } f_1 = 3^{e_2}, \ f_0 = 2^{e_1}.$$

The constraint of producing a product from two consecutive integers means that either:

$$c = f_1 \text{ with } c-1 = f_0 \text{ or } c = f_0 \text{ with } c-1 = f_1.$$

In general, the partitions of consideration for $P_2(c)$ of order r will number half the total 2^r partitions because of symmetry, and then minus one of those for the identity case, leaving $2^{r-1} - 1$. For each partition of $P_2(c)$, and for each order r, solutions for c can be found or, eliminated, by solving the resulting Diophantine equations over all prime factor exponents, e_i. Some of the resulting equations will have solutions for c, and some will not.

A.1. Base Cases $r = 2$ and $r = 3$

Theorem 1 (Order 2 Base Case)**.** *For the Diophantine equation* $c \times (c-1) = 2^{e_1} \times 3^{e_2}$ *where* $e_1, e_2 > 0$*, the only solutions are* $c \in \ \{3, 4, 9\}$*.*

Proof. Since c and $c-1$ are coprime, their prime factorizations must be disjoint. This leaves two cases to consider:

Case 1: Let $c = 2^{e_1}$ and $c-1 = 3^{e_2}$. Then:

$$2^{e_1} - 1 = 3^{e_2}.$$

Starting with $e_1 = 1$ the equation can be examined for a possible solution in e_2. For $e_1 = 2$ this will be satisfied with $e_2 = 1$, for $c = 4$.

Case 2: Let $c = 3^{e_2}$ and $c-1 = 2^{e_1}$. Then:

$$3^{e_2} - 2^{e_1} = 1$$

, which has a solution for $e_2 = 1$ at $e_1 = 1$ where $c = 3$. By Mihăilescu's theorem (= the former Catalan conjecture)[8], the only non-trivial perfect power difference between consecutive terms of recurrence sequences is $9 = 3^2$ and $8 = 2^3$. This yields the solution $c = 9$ with no possible solutions beyond $c = 9$.

Therefore, the complete set of solutions is $c \in \{3, 4, 9\}$. $\square$

Moving on to $r = 3$:

Theorem 2 (Order 3 Base Case)**.** *For the Diophantine equation $c \times (c-1) = 2^{e_1} \times 3^{e_2} \times 5^{e_3}$ where $e_1, e_2, e_3 > 0$, there exist no solutions for $c > 81$.*

Proof. The coprimality of c and $c-1$ requires disjoint prime factorizations. Consider the possible partitions:

$$\begin{aligned} &\text{Case 1:} \quad c = 2^{e_1},\ (c-1) = 3^{e_2} \times 5^{e_3}. \\ &\text{Case 2:} \quad c = 3^{e_2},\ (c-1) = 2^{e_1} \times 5^{e_3}. \\ &\text{Case 3:} \quad c = 5^{e_3},\ (c-1) = 2^{e_1} \times 3^{e_2}. \end{aligned}$$

and their reverses.

For $c > 81$:

- In Case 1, $2^{e_1} - 1 = 3^{e_2} \times 5^{e_3}$. For $e_1 \geq 7$, the growth rate of 2^{e_1} makes this impossible.
- In Case 2, $3^{e_2} - 1 = 2^{e_1} \times 5^{e_3}$. For $e_2 \geq 5$, the difference between consecutive powers of 3 grows too rapidly.
- In Case 3, $5^{e_3} - 1 = 2^{e_1} \times 3^{e_2}$. Similar growth rate arguments apply.

The reverse cases follow by the same arguments. The complete set of solutions is $c \in \{10, 16, 25, 81\}$. $\square$

A.2. Induction from Base Cases

Having shown the process for cases $r = 2$ and $r = 3$, the cases for $r > 3$ can be shown by induction:

Theorem 3 (All Primorial Complete Orders have Maximum $P_2(c)$)**.** *For any order r of π-complete products of consecutive integers $c \times (c-1)$, there exists a maximum value M_r such that no solutions exist for $c > M_r$.*

Proof. This proof proceeds by induction on the order r. Base Cases: It has been shown that $M_2 = 9$ and $M_3 = 81$. Inductive Step: Assume that for some order $r \geq 3$, there exists a maximum value M_r beyond which no solutions exist for:

$$c \times (c-1) = 2^{e_1} \times 3^{e_2} \times \cdots \times p_r^{e_r}.$$

For order $r+1$, adding prime p_{r+1}:

$$c \times (c-1) = 2^{e_1} \times 3^{e_2} \times \cdots \times p_r^{e_r} \times p_{r+1}^{e_{r+1}}.$$

The following constraints ensure a bound exists:

1. Growth Rate: The product $c \times (c-1)$ grows as $O(c^2)$, while adding p_{r+1} increases the minimal gap between possible factorizations.

2. Quantization: Each prime factor must be assigned entirely to either c or $c-1$. Adding p_{r+1} makes the partitioning more restrictive, not less.

3. Prime Power Differences: For large c, consecutive integers become relatively closer:
$$\frac{c}{c-1} \to 1 \text{ as } c \to \infty.$$

This forces increasingly precise alignment of prime factorizations, which becomes impossible with additional prime factors. Therefore, if M_r exists for order r, then M_{r+1} must exist for order $r+1$. □

The following tables show the complete solutions for cases $r = 4$ through $r = 8$:

$r\#m\ P_2(c)$	Factorization	Square Root Limitation
$4\#1 = P_2(15)$	$2 \times 3 \times 5 \times 7$	$0.5 - 0.4914 = 0.0086 < \frac{1}{14}$
$4\#2 = P_2(21)$	$2^2 \times 3 \times 5 \times 7$	$0.5 - 0.4939 = 0.0061 < \frac{1}{20}$
$4\#6 = P_2(36)$	$2^2 \times 3^2 \times 5 \times 7$	$0.5 - 0.4965 = 0.0035 < \frac{1}{35}$
$4\#75 = P_2(126)$	$2 \times 3^2 \times 5^3 \times 7$	$0.5 - 0.4990 = 0.001 < \frac{1}{125}$
$4\#240 = P_2(225)$	$2^5 \times 3^2 \times 5^2 \times 7$	$0.5 - 0.4994 = 0.0006 < \frac{1}{224}$
$4\#27{,}440 = P_2(2{,}401)$	$2^5 \times 3 \times 5^2 \times 7^4$	$0.5 - 0.4999 = 0.0005167 < \frac{1}{2{,}400}$
$4\#91{,}125 = P_2(4{,}375)$	$2 \times 3^7 \times 5^4 \times 7$	$0.5 - 0.49997 = 0.00003 < \frac{1}{4{,}374}$

Table 6: $\mathcal{P}(4)$ Square Root Solutions

$r\#m\ P_2(c)$	Factorization	Square Root Limitation
$5\#64 = P_2(385)$	$2^7 \times 3 \times 5 \times 7 \times 11$	$0.5 - 0.49967 = 0.00033 < \frac{1}{384}$
$5\#84 = P_2(441)$	$2^3 \times 3^2 \times 5 \times 7^2 \times 11$	$0.5 - 0.4997 = 0.000284 < \frac{1}{440}$
$5\#126 = P_2(540)$	$2^2 \times 3^3 \times 5 \times 7^2 \times 11$	$0.5 - 0.499768 = 0.000232 < \frac{1}{539}$
$5\#3{,}960 = P_2(3{,}025)$	$2^4 \times 3^3 \times 5^2 \times 7 \times 11^2$	$0.5 - 0.4999587 = 0.0000413 < \frac{1}{3{,}024}$
$5\#41{,}580 = P_2(9{,}801)$	$2^3 \times 3^4 \times 5^2 \times 7^2 \times 11^2$	$0.5 - 0.499987 = 0.000013 < \frac{1}{9{,}800}$

Table 7: $\mathcal{P}(5)$ Square Root Solutions

$r\#m\ P_2(c)$	Factorization	Square Root Limitation
$6\#98 = P_2(1{,}716)$	$2^2 \times 3 \times 5 \times 7^3 \times 11 \times 13$	$0.5 - 0.49992713 = 0.00007287 < \frac{1}{1{,}715}$
$6\#144 = P_2(2{,}080)$	$2^5 \times 3^3 \times 5 \times 7 \times 11 \times 13$	$0.5 - 0.49993989 = 0.00006011 < \frac{1}{2{,}079}$

Table 8: $\mathcal{P}(6)$ Square Root Solutions

$r\#m\ P_2(c)$	Factorization	Square Root Limitation
$7\#1 = P_2(715)$	$2 \times 3 \times 5 \times 7 \times 11 \times 13 \times 17$	$0.5 - 0.499825 = 0.000175 < \frac{1}{714}$
$7\#3{,}300 = P_2(12{,}376)$	$2^3 \times 3^2 \times 5^3 \times 7 \times 11 \times 13 \times 17$	$0.5 - 0.4999899 = 0.0000101 < \frac{1}{12{,}375}$
$7\#74{,}088 = P_2(94{,}481)$	$2^4 \times 3^4 \times 5 \times 7^4 \times 11 \times 13 \times 17$	$0.5 - 0.49999935 = 0.00000065 < \frac{1}{194{,}480}$

Table 9: $\mathcal{P}_c(7)$ Square Root Solutions

$r\#m\ P_2(c)$	Factorization	Square Root Limitation
$8\#41{,}382 = P_2(633{,}556)$	$2^2 \times 3^3 \times 5 \times 7 \times 11^3 \times 13 \times 17 \times 19^2$	$0.5 - 0.4999998 = 0.0000002 < \frac{1}{633{,}555}$

Table 10: $\mathcal{P}_c(8)$ Square Root Solutions

A.3. Diopantine Equations of Partitions

The partitioning solutions of the π-complete products into consecutive integers, c and $c-1$, can be expressed as equations of the differences of multiplicative groups of prime factors that equal 1:

In partition $S_{1,1} \rightarrow$	$(2^1) - (1) = 1$
In partition $S_{2,1} \rightarrow$	$(2^2) - (3^1) = 1$
In partition $S_{2,2} \rightarrow$	$(3^1) - (2^1) = 1$
In partition $S_{2,2} \rightarrow$	$(3^2) - (2^3) = 1$
In partition $S_{3,1} \rightarrow$	$(2^4) - (3^1 \times 5^1) = 1$
In partition $S_{3,2} \rightarrow$	$(3^4) - (2^4 \times 5^1) = 1$
In partition $S_{3,3} \rightarrow$	$(2^1 \times 3^1) - (5^1) = 1$
In partition $S_{3,4} \rightarrow$	$(5^2) - (2^3 \times 3^1) = 1$
In partition $S_{3,5} \rightarrow$	$(2^1 \times 5^1) - (3^2) = 1$
In partition $S_{4,3} \rightarrow$	$(2^2 \times 3^2) - (5^1 \times 7^1) = 1$
In partition $S_{4,6} \rightarrow$	$(3^1 \times 5^1) - (2^2 \times 7^1) = 1$
In partition $S_{4,6} \rightarrow$	$(3^2 \times 5^2) - (2^5 \times 7^1) = 1$
In partition $S_{4,7} \rightarrow$	$(2^1 \times 3^7 \times 5^4) - (7^1) = 1$
In partition $S_{4,7} \rightarrow$	$(2^5 \times 3^1 \times 5^2) - (7^4) = 1$
In partition $S_{4,10} \rightarrow$	$(3^1 \times 7^1) - (2^2 \times 5^1) = 1$
In partition $S_{4,11} \rightarrow$	$(2^1 \times 3^2 \times 7^1) - (5^3) = 1$
In partition $S_{5,7} \rightarrow$	$(2^2 \times 3^3 \times 5^1) - (7^2 \times 11^1) = 1$
In partition $S_{5,7} \rightarrow$	$(2^3 \times 3^4 \times 5^2) - (7^2 \times 11^2) = 1$
In partition $S_{5,7} \rightarrow$	$(2^4 \times 3^3 \times 5^2) - (7^1 \times 11^2) = 1$
In partition $S_{5,10} \rightarrow$	$(3^2 \times 7^2) - (2^3 \times 5^1 \times 11^1) = 1$
In partition $S_{5,28} \rightarrow$	$(5^1 \times 7^1 \times 11^1) - (2^7 \times 3^1) = 1$
In partition $S_{6,7} \rightarrow$	$(2^5 \times 3^3 \times 5^1) - (7^1 \times 11^1 \times 13^1) = 1$
In partition $S_{6,51} \rightarrow$	$(2^2 \times 3^1 \times 11^1 \times 13^1) - (5^1 \times 7^3) = 1$
In partition $S_{7,7} \rightarrow$	$(2^3 \times 3^2 \times 5^3) - (7^1 \times 11^1 \times 13^1 \times 17^1) = 1$
In partition $S_{7,7} \rightarrow$	$(2^4 \times 3^4 \times 5^1) - (7^4 \times 11^1 \times 13^1 \times 17^1) = 1$
In partition $S_{7,52} \rightarrow$	$(5^1 \times 11^1 \times 13^1) - (2^1 \times 3^1 \times 7^1 \times 17^1) = 1$
In partition $S_{8,31} \rightarrow$	$(2^2 \times 3^3 \times 5^1 \times 7^1 \times 11^3) - (13^1 \times 17^1 \times 19^2) = 1$

These solutions have no prime with power greater than seven, in accord with the theory that larger values makes the quantization probem cut off more solutions. This is also consistant with the theory that higher π-complete orders with more prime multipliers operate likewise, and require lower powers. As order seven has no solution with any prime power greater than four, and order eight has no solution with any prime power greater than three, one might expect orders nine and above to be even more limited in solutions, if any exist at all.

A.4. No Solutions Beyond Order 8

The final proof in this section addresses the impossibility of π-complete products of pairs if consecutive integers to exist for $\mathcal{P}(9)$ and above.

Theorem 4 (No Consecutive Integer Product for $r \geq 9$)**.** *Let $r \geq 9$ and consider the product of two consecutive integers*

$$P_2(c) \;=\; c \times (c-1).$$

If $P_2(c)$ is $\in \mathcal{P}_c(r)$, meaning

$$P_2(c) \;=\; 2^{e_1} \times 3^{e_2} \times 5^{e_3} \times \cdots \times p_r^{e_r} \quad (\, e_i > 0\,),$$

then no such integer c satisfies this equation for $r \geq 9$. Equivalently, there are no *new solutions c for consecutive-integer π-complete products once the 9th prime (which is 23) is introduced.*

Proof. **Step 1: Review of the Base Cases** ($r \leq 8$)**.** From the prior explicit searches and arguments, it can be shown that:

- For orders $r = 2$ through $r = 8$, there are *finitely many* integer values c for which $c \times (c-1)$ is $\in \mathcal{P}_c(r)$.
- Concretely, once $r = 8$ is reached (the largest prime involved is 19), the *last* consecutive product found is

  $$c = 633{,}556, \quad c \times (c-1) \;=\; 2^2 \times 3^3 \times 5^1 \times 7^1 \times 11^3 \times 13^1 \times 17^1 \times 19^2.$$

  No further doublets appear for any larger c at order $r = 8$.

Thus, up to and including $r = 8$, the exhaustive analysis shows that the highest possible c is 633,556, and *no larger* consecutive integers can produce an order-8 π-complete product.

Step 2: Introduction of the 9th Prime $p_9 = 23$. Suppose, for contradiction, that there exists

$$c > 1 \quad \text{such that} \quad c \times (c-1) \;=\; 2^{e_1} \times 3^{e_2} \times \cdots \times p_9^{e_9},$$

where $p_9 = 23$ and $e_i > 0$. Since c and $c-1$ are coprime, the prime 23 (with exponent $e_9 \geq 1$) must lie wholly on one factor or the other:

1. **Case A:** $c = 23^{e_9} \times M, \quad (c-1) = N, \quad \gcd(M,N) = 1, \quad \gcd(M,23) = 1 = \gcd(N,23)$.
2. **Case B:** $(c-1) = 23^{e_9} \times N, \quad c = M, \quad \gcd(M,N) = 1, \quad \gcd(M,23) = 1 = \gcd(N,23)$.

Either way, this is a Diophantine equation of the form

$$\big(\text{prime product to 23}\big)_{\text{going to}}\, c \;-\; \big(\text{remaining to 23}\big)_{\text{going to}}\, (c-1) \;=\; 1.$$

As soon as 23 appears, the gap needed to "fit" an integer squarely between two prime-power products becomes significantly narrower; in fact, the *Square Root Limitation* argument (Lemma 18) shows that

$$\frac{2c-1}{2} - \sqrt{c(c-1)} \;<\; \frac{1}{c-1},$$

and for large c (well below the tens or hundreds of thousands), no arrangement of prime exponents up to and including 23 can stay fully π-complete while still matching the difference of only 1.

Moreover, extensive numeric checks for $r = 9$ confirm that no new consecutive-integer product with prime 23 included can appear beyond the previously discovered cutoff for $r = 8$. This is consistent with the observation that *each time* a new prime p_{r+1} is introduced, it creates an additional "partition" demand on c and $(c-1)$. But for large c, the difference 1 is far too constraining to allow a fresh solution.

Step 3: Inductive Extension to All $r > 9$. Now, the inductive argument is that if consecutive *doublet* products fail to be π-complete upon introducing prime $p_9 = 23$, then they must also fail for every larger prime p_r with $r > 9$:

- Induction Hypothesis: Assume that for some $\mathcal{P}(r)$ with $r \geq 9$, there is no integer c making $c \times (c-1)$ incorporate p_r with exponent ≥ 1 (together with all smaller primes) while still equaling a consecutive product.

- Inductive Step: Passing from r to $r+1$ introduces a new largest prime $p_{r+1} > p_r$. This strictly increases both the minimal product of all primes up to p_{r+1} (the $(r+1)$-th primorial) and the difficulty of allocating prime exponents so that c and $c-1$ remain coprime and differ by exactly 1. But by the *Square Root Limitation* (and the mismatch in "consecutive partitioning"), allowing an even bigger prime means even *fewer* ways to satisfy the difference-of-1 constraint. Consequently, no new solutions appear at order $r+1$.

Hence by induction, once the threshold $r = 9$ is crossed, **no** solutions exist for any higher order.

□

A.5. Conclusion

It is interesting to note that as the number of partitions increases for $\mathcal{P}_c(r)$ from $r = 2$ to $r = 4$, so does the number of solutions for c, however, the number of solutions generally decreases after order 4, even though the number of partitions continues to increase exponentially. This is consistent with the increasing restriction of possible combinations of integer exponents of the prime factors of $P_2(c)$ while keeping those products π-complete.

The change in the divisibility of products of consecutive integers by primorial numbers can be seen in examples presented in Appendix D. Here, the first 25 products of consecutive integers that are divisible by primorials 7# through 11# are shown, with π-complete products underlined. The last π-complete product of consective integers is divisible by 8#, as underlined in that section. What is particularly of note, in each section, is that in the prime factorizations of the quotients, the top prime factor is always so much higher than the top prime factor of the primorial, except for the special case in 8#. Going up from 8#, this difference continues as the top prime factor of the primorial increases.

While the demonstrations in this appendix deal with the $P_2(c)$ products, the results are directly extensible to $P_3(c)$ through $P_5(c)$, as explained in the main body of the proof of the ABC Factorial Conjecture. As $P_2(c)$ has the largest maximum c encounetered, it is the most significant case in the general proof, and as such, was chosen as the subject of this appendix for careful study.

B. Anti-sieve Visualization

A useful visualization can come from turning the ancient Sieve of Eratosthenes inside out to make the “Anti-sieve.” In the Anti-sieve concept, instead of primes skipping up the number line knocking out composite numbers where they land, the primes mark those numbers for preservation. If, after all primes less than or equal to a number have skipped by, a consecutive pair of numbers is preserved as π-complete if they have a complete set of marks between them (however, those will be different marks because they are coprime). In this visualization, the property of being coprime is clear, because if a prime lands on the lesser number, it cannot hit the next number because it has to jump at least 2. If interest is in the product of three numbers, the picture is, for the most part, the same because the lowest and highest could only share the prime factor 2, and must be relatively prime for all the others.

The circumstance that reduces the rate of π-complete products as the pair, or k-tuple, moves up the number line is that primes larger than their current largest prime, by more than one prime number index, will come along and make a mark that then leaves a space ahead of what was the largest prime factor. If one imagines these primes skipping along in order of magnitude, then a lower prime is not going to come along and mark that empty space. Thus, as numbers or pairs of numbers go up the number line, they tend to be dominated by missing primes in their prime factorizations (as are all prime numbers past 2) and the relative percentage of π-complete numbers drops (but never goes to zero).[6]

A striking example occurs at the consecutive pair

$$(633{,}556,\ 633{,}555).$$

On the surface, one might imagine that among the tens of thousands of primes below 633,555, many different primes would “land on” or divide one of these integers, thus forcing its product to be missing some lower prime and **not** be π-complete. Yet, if factored individually:

$$633{,}555 \;=\; 3^3 \,\times\, 5^1 \,\times\, 13^1 \,\times\, 19^2 \;\text{ and }\; 633{,}556 \;=\; 2^2 \,\times\, 7^1 \,\times\, 11^3 \,\times\, 17^1$$

so one sees only eight distinct primes:

$$\{2, 3, 5, 7, 11, 13, 17, 19\}.$$

In other words, while **each** prime up to and including 19 **must** land on one or the other in the pair, **no** other prime up to 633,555, of which there are roughly 50,000 by the prime number theorem[1,9], ever “hits” these two integers. Thus their product,

$$633{,}555 \times 633{,}556 \;=\; 2^2 \times 3^3 \times 5 \times 7 \times 11^3 \times 13 \times 17 \times 19^2$$

includes all primes from 2 to 19 and avoids entirely every prime > 23. From the Anti-sieve viewpoint, it is as though each new prime $p > 23$ jumped over both 633,555 and 633,556, leaving them unmarked and so preserving a contiguous block of small primes in their factorization. This is precisely why the pair

$$(633{,}556, \times 633{,}555)$$

is the last π-complete consecutive integer product as **no** larger pair can so effectively evade all those intermediate primes.

C. Factorials as π-complete Numbers

Since all factorial numbers are π-complete, they can be expressed as a product of a base primorial and a multiplier that has no greater prime factor than the base primorial. It is interesting to compare the slow rate of growth by the primorial base to the rapid growth of the multiplier needed to keep up with the factorials, as shown in Table 11.

n	$n!$	$r\#m$ form
1	1	1#1
2	2	2#1
3	6	2#3
4	24	3#4
5	120	3#20
6	720	3#120
7	5,040	4#210
8	40,320	4#1,680
9	362,880	4#15,120
10	3,628,800	4#151,200
11	39,916,800	5#1,663,200
12	479,001,600	5#19,958,400
13	6,227,020,800	5#259,459,200
14	87,178,291,200	5#3,632,428,800
15	1,307,674,368,000	6#54,486,432,000
16	20,922,789,888,000	6#871,782,912,000
17	355,687,428,096,000	7#14,820,309,504,000
18	6,402,373,705,728,000	7#266,765,571,072,000
19	121,645,100,408,832,000	7#5,068,545,850,368,000
20	2,432,902,008,176,640,000	8#101,370,917,007,360,000

Table 11: First 20 factorials expressed in $r\#m$ form

D. Consecutive Integer Products Divisible by Primorials

First 25 products of consecutive integers divisible by 7 primorial:

$$
\begin{aligned}
\underline{715 \times 714} \div 7\# &= 1 \\
9{,}009 \times 9{,}010 \div 7\# &= 159 \;=\; 3 \times 53 \\
11{,}934 \times 11{,}935 \div 7\# &= 279 \;=\; 3^2 \times 31 \\
\underline{12{,}375 \times 12{,}376} \div 7\# &= 300 \;=\; 2^2 \times 3 \times 5^2 \\
13{,}650 \times 13{,}651 \div 7\# &= 365 \;=\; 5 \times 73 \\
34{,}034 \times 34{,}035 \div 7\# &= 2{,}269 \\
37{,}400 \times 37{,}401 \div 7\# &= 2{,}740 \;=\; 2^2 \times 5 \times 137 \\
38{,}675 \times 38{,}676 \div 7\# &= 2{,}930 \;=\; 2 \times 5 \times 293 \\
46{,}409 \times 46{,}410 \div 7\# &= 4{,}219 \\
47{,}124 \times 47{,}125 \div 7\# &= 4{,}350 \;=\; 2 \times 3 \times 5^2 \times 29 \\
47{,}684 \times 47{,}685 \div 7\# &= 4{,}454 \;=\; 2 \times 17 \times 131 \\
51{,}050 \times 51{,}051 \div 7\# &= 5{,}105 \;=\; 5 \times 1{,}021 \\
51{,}765 \times 51{,}766 \div 7\# &= 5{,}249 \;=\; 29 \times 181 \\
58{,}344 \times 58{,}345 \div 7\# &= 6{,}668 \;=\; 2^2 \times 1{,}667 \\
60{,}060 \times 60{,}061 \div 7\# &= 7{,}066 \;=\; 2 \times 3{,}533 \\
60{,}774 \times 60{,}775 \div 7\# &= 7{,}235 \;=\; 5 \times 1{,}447 \\
62{,}985 \times 62{,}986 \div 7\# &= 7{,}771 \;=\; 19 \times 409 \\
71{,}994 \times 71{,}995 \div 7\# &= 10{,}153 \;=\; 11 \times 13 \times 71 \\
85{,}085 \times 85{,}086 \div 7\# &= 14{,}181 \;=\; 3 \times 29 \times 163 \\
85{,}799 \times 85{,}800 \div 7\# &= 14{,}420 \;=\; 2^2 \times 5 \times 7 \times 103 \\
94{,}094 \times 94{,}095 \div 7\# &= 17{,}343 \;=\; 3^2 \times 41 \times 47 \\
97{,}019 \times 97{,}020 \div 7\# &= 18{,}438 \;=\; 2 \times 3 \times 7 \times 439 \\
97{,}460 \times 97{,}461 \div 7\# &= 18{,}606 \;=\; 2 \times 3 \times 7 \times 443 \\
98{,}175 \times 98{,}176 \div 7\# &= 18{,}880 \;=\; 2^6 \times 5 \times 59 \\
98{,}735 \times 98{,}736 \div 7\# &= 19{,}096 \;=\; 2^3 \times 7 \times 11 \times 31
\end{aligned}
$$

The last π-complete product divisible by 7# is at:

$$\underline{194{,}480 \times 194{,}481} \div 7\# = 74{,}088 \;=\; 2^3 \times 3^3 \times 7^3$$

First 25 products of consecutive integers divisible by 8 primorial:

$$\begin{aligned}
62{,}985 \times 62{,}986 \div 8\# &= 409\\
157{,}794 \times 157{,}795 \div 8\# &= 2{,}567 \;=\; 17 \times 151\\
203{,}489 \times 203{,}490 \div 8\# &= 4{,}269 \;=\; 3 \times 1{,}423\\
217{,}854 \times 217{,}855 \div 8\# &= 4{,}893 \;=\; 3 \times 7 \times 233\\
241{,}604 \times 241{,}605 \div 8\# &= 6{,}018 \;=\; 2 \times 3 \times 17 \times 59\\
293{,}930 \times 293{,}931 \div 8\# &= 8{,}907 \;=\; 3 \times 2{,}969\\
307{,}020 \times 307{,}021 \div 8\# &= 9{,}718 \;=\; 2 \times 43 \times 113\\
352{,}715 \times 352{,}716 \div 8\# &= 12{,}826 \;=\; 2 \times 11^2 \times 53\\
367{,}080 \times 367{,}081 \div 8\# &= 13{,}892 \;=\; 2^2 \times 23 \times 151\\
510{,}510 \times 510{,}511 \div 8\# &= 26{,}869 \;=\; 97 \times 277\\
570{,}570 \times 570{,}571 \div 8\# &= 33{,}563\\
573{,}495 \times 573{,}496 \div 8\# &= 33{,}908 \;=\; 2^2 \times 7^2 \times 173\\
\underline{633{,}555 \times 633{,}556} \div 8\# &= 41{,}382 = 2 \times 3^2 \times 11^2 \times 19\\
728{,}364 \times 728{,}365 \div 8\# &= 54{,}694 \;=\; 2 \times 23 \times 29 \times 41\\
752{,}114 \times 752{,}115 \div 8\# &= 58{,}319 \;=\; 29 \times 2{,}011\\
766{,}479 \times 766{,}480 \div 8\# &= 60{,}568 \;=\; 2^3 \times 67 \times 113\\
804{,}440 \times 804{,}441 \div 8\# &= 66{,}716 \;=\; 2^2 \times 13 \times 1{,}283\\
812{,}174 \times 812{,}175 \div 8\# &= 68{,}005 \;=\; 5 \times 7 \times 29 \times 67\\
864{,}500 \times 864{,}501 \div 8\# &= 77{,}050 \;=\; 2 \times 5^2 \times 23 \times 67\\
877{,}590 \times 877{,}591 \div 8\# &= 79{,}401 \;=\; 3 \times 7 \times 19 \times 199\\
969{,}969 \times 969{,}970 \div 8\# &= 96{,}997\\
1{,}032{,}954 \times 1{,}032{,}955 \div 8\# &= 110{,}003 \;=\; 41 \times 2{,}683\\
1{,}081{,}080 \times 1{,}081{,}081 \div 8\# &= 120{,}492 \;=\; 2^2 \times 3^2 \times 3{,}347\\
1{,}119{,}195 \times 1{,}119{,}196 \div 8\# &= 129{,}138 \;=\; 2 \times 3 \times 21{,}523\\
1{,}144{,}065 \times 1{,}144{,}066 \div 8\# &= 134{,}941 \;=\; 23 \times 5{,}867
\end{aligned}$$

The lowest top prime found in the multiplier was 19, at the last π-complete case.

First 25 products of consecutive integers divisible by 9 primorial:

$$\begin{aligned}
367{,}080 \times 367{,}081 \div 9\# &= 604 \ = \ 2^2 \times 151\\
728{,}364 \times 728{,}365 \div 9\# &= 2{,}378 \ = \ 2 \times 29 \times 41\\
864{,}500 \times 864{,}501 \div 9\# &= 3{,}350 \ = \ 2 \times 5^2 \times 67\\
1{,}144{,}065 \times 1{,}144{,}066 \div 9\# &= 5{,}867\\
1{,}322{,}684 \times 1{,}322{,}685 \div 9\# &= 7{,}842 \ = \ 2 \times 3 \times 1{,}307\\
1{,}540{,}539 \times 1{,}540{,}540 \div 9\# &= 10{,}638 \ = \ 2 \times 3^3 \times 197\\
2{,}187{,}185 \times 2{,}187{,}186 \div 9\# &= 21{,}443 \ = \ 41 \times 523\\
2{,}466{,}750 \times 2{,}466{,}751 \div 9\# &= 27{,}275 \ = \ 5^2 \times 1{,}091\\
5{,}616{,}324 \times 5{,}616{,}325 \div 9\# &= 141{,}390 \ = \ 2 \times 3^2 \times 5 \times 1{,}571\\
6{,}032{,}025 \times 6{,}032{,}026 \div 9\# &= 163{,}095 \ = \ 3 \times 5 \times 83 \times 131\\
6{,}113{,}744 \times 6{,}113{,}745 \div 9\# &= 167{,}544 \ = \ 2^3 \times 3^2 \times 13 \times 179\\
6{,}393{,}309 \times 6{,}393{,}310 \div 9\# &= 183{,}217 \ = \ 19 \times 9{,}643\\
7{,}039{,}955 \times 7{,}039{,}956 \div 9\# &= 222{,}154 \ = \ 2 \times 277 \times 401\\
7{,}050{,}120 \times 7{,}050{,}121 \div 9\# &= 222{,}796 \ = \ 2^2 \times 7 \times 73 \times 109\\
7{,}354{,}710 \times 7{,}354{,}711 \div 9\# &= 242{,}463 \ = \ 3 \times 13 \times 6{,}217\\
7{,}436{,}429 \times 7{,}436{,}430 \div 9\# &= 247{,}881 \ = \ 3 \times 53 \times 1{,}559\\
7{,}715{,}994 \times 7{,}715{,}995 \div 9\# &= 266{,}869 \ = \ 23 \times 41 \times 283\\
7{,}803{,}509 \times 7{,}803{,}510 \div 9\# &= 272{,}957 \ = \ 107 \times 2{,}551\\
7{,}852{,}130 \times 7{,}852{,}131 \div 9\# &= 276{,}369 \ = \ 3 \times 17 \times 5{,}419\\
8{,}083{,}074 \times 8{,}083{,}075 \div 9\# &= 292{,}865 \ = \ 5 \times 58{,}573\\
8{,}219{,}210 \times 8{,}219{,}211 \div 9\# &= 302{,}813 \ = \ 7 \times 181 \times 239\\
8{,}580{,}494 \times 8{,}580{,}495 \div 9\# &= 330{,}019\\
8{,}895{,}249 \times 8{,}895{,}250 \div 9\# &= 354{,}675 \ = \ 3 \times 5^2 \times 4{,}729\\
8{,}947{,}575 \times 8{,}947{,}576 \div 9\# &= 358{,}860 \ = \ 2^2 \times 3 \times 5 \times 5{,}981\\
9{,}541{,}895 \times 9{,}541{,}896 \div 9\# &= 408{,}116 \ = \ 2^2 \times 257 \times 397
\end{aligned}$$

The lowest top prime found in the multiplier was 41, which is much larger than 23, so no π-complete products.

First 25 products of consecutive integers divisible by 10 primorial:

$$
\begin{aligned}
728{,}364 \times 728{,}365 \div 10\# &= 82 \;=\; 2 \times 41\\
12{,}299{,}364 \times 12{,}299{,}365 \div 10\# &= 23{,}382 \;=\; 2 \times 3^3 \times 433\\
14{,}612{,}520 \times 14{,}612{,}521 \div 10\# &= 33{,}004 \;=\; 2^2 \times 37 \times 223\\
18{,}122{,}390 \times 18{,}122{,}391 \div 10\# &= 50{,}763 \;=\; 3 \times 16{,}921\\
28{,}881{,}215 \times 28{,}881{,}216 \div 10\# &= 128{,}928 \;=\; 2^5 \times 3 \times 17 \times 79\\
30{,}421{,}754 \times 30{,}421{,}755 \div 10\# &= 143{,}049 \;=\; 3 \times 41 \times 1{,}163\\
30{,}788{,}835 \times 30{,}788{,}836 \div 10\# &= 146{,}522 \;=\; 2 \times 61 \times 1{,}201\\
36{,}815{,}064 \times 36{,}815{,}065 \div 10\# &= 209{,}492 \;=\; 2^2 \times 83 \times 631\\
44{,}222{,}100 \times 44{,}222{,}101 \div 10\# &= 302{,}270 \;=\; 2 \times 5 \times 167 \times 181\\
45{,}401{,}355 \times 45{,}401{,}356 \div 10\# &= 318{,}606 \;=\; 2 \times 3 \times 53{,}101\\
64{,}822{,}394 \times 64{,}822{,}395 \div 10\# &= 649{,}481 \;=\; 7 \times 31 \times 41 \times 73\\
75{,}010{,}935 \times 75{,}010{,}936 \div 10\# &= 869{,}692 \;=\; 2^2 \times 41 \times 5{,}303\\
75{,}823{,}110 \times 75{,}823{,}111 \div 10\# &= 888{,}627 \;=\; 3 \times 139 \times 2{,}131\\
95{,}244{,}149 \times 95{,}244{,}150 \div 10\# &= 1{,}402{,}145 \;=\; 5 \times 193 \times 1{,}453\\
103{,}830{,}440 \times 103{,}830{,}441 \div 10\# &= 1{,}666{,}348 \;=\; 2^2 \times 619 \times 673\\
105{,}432{,}690 \times 105{,}432{,}691 \div 10\# &= 1{,}718{,}173 \;=\; 17 \times 211 \times 479\\
110{,}223{,}750 \times 110{,}223{,}751 \div 10\# &= 1{,}877{,}875 \;=\; 5^3 \times 83 \times 181\\
113{,}733{,}620 \times 113{,}733{,}621 \div 10\# &= 1{,}999{,}374 \;=\; 2 \times 3 \times 13 \times 25{,}633\\
126{,}032{,}984 \times 126{,}032{,}985 \div 10\# &= 2{,}455{,}188 \;=\; 2^2 \times 3 \times 204{,}599\\
128{,}346{,}140 \times 128{,}346{,}141 \div 10\# &= 2{,}546{,}138 \;=\; 2 \times 7^2 \times 25{,}981\\
133{,}097{,}964 \times 133{,}097{,}965 \div 10\# &= 2{,}738{,}162 \;=\; 2 \times 7 \times 131 \times 1{,}493\\
133{,}440{,}020 \times 133{,}440{,}021 \div 10\# &= 2{,}752{,}254 \;=\; 2 \times 3^2 \times 107 \times 1{,}429\\
139{,}833{,}330 \times 139{,}833{,}331 \div 10\# &= 3{,}022{,}301 \;=\; 131 \times 23{,}071\\
140{,}645{,}505 \times 140{,}645{,}506 \div 10\# &= 3{,}057{,}511 \;=\; 1{,}447 \times 2{,}113\\
141{,}373{,}869 \times 141{,}373{,}870 \div 10\# &= 3{,}089{,}261 \;=\; 7 \times 173 \times 2{,}551
\end{aligned}
$$

The lowest top prime found in the multiplier was 41, which is much larger than 29, so no π-complete products.

First 24 products of consecutive integers divisible by 11 primorial:

$$
\begin{aligned}
64{,}822{,}394 \times 64{,}822{,}395 \div 11\# &= 20{,}951 \;=\; 7 \times 41 \times 73\\
158{,}767{,}895 \times 158{,}767{,}896 \div 11\# &= 125{,}684 \;=\; 2^2 \times 13 \times 2{,}417\\
171{,}434{,}340 \times 171{,}434{,}341 \div 11\# &= 146{,}538 \;=\; 2 \times 3^2 \times 7 \times 1{,}163\\
283{,}988{,}705 \times 283{,}988{,}706 \div 11\# &= 402{,}121 \;=\; 163 \times 2{,}467\\
443{,}998{,}554 \times 443{,}998{,}555 \div 11\# &= 982{,}919 \;=\; 7 \times 140{,}417\\
663{,}164{,}865 \times 663{,}164{,}866 \div 11\# &= 2{,}192{,}793 \;=\; 3 \times 389 \times 1{,}879\\
727{,}987{,}260 \times 727{,}987{,}261 \div 11\# &= 2{,}642{,}422 \;=\; 2 \times 29^2 \times 1{,}571\\
862{,}239{,}455 \times 862{,}239{,}456 \div 11\# &= 3{,}706{,}896 \;=\; 2^4 \times 3 \times 29 \times 2{,}663\\
886{,}755{,}155 \times 886{,}755{,}156 \div 11\# &= 3{,}920{,}686 \;=\; 2 \times 7^2 \times 11 \times 3{,}637\\
976{,}359{,}384 \times 976{,}359{,}385 \div 11\# &= 4{,}753{,}068 \;=\; 2^2 \times 3 \times 431 \times 919\\
983{,}038{,}055 \times 983{,}038{,}056 \div 11\# &= 4{,}818{,}316 \;=\; 2^2 \times 23 \times 83 \times 631\\
1{,}047{,}860{,}450 \times 1{,}047{,}860{,}451 \div 11\# &= 5{,}474{,}715 \;=\; 3 \times 5 \times 179 \times 2{,}039\\
1{,}306{,}238{,}010 \times 1{,}306{,}238{,}011 \div 11\# &= 8{,}507{,}447 \;=\; 13 \times 23 \times 37 \times 769\\
1{,}457{,}458{,}365 \times 1{,}457{,}458{,}366 \div 11\# &= 10{,}591{,}243 \;=\; 13 \times 31 \times 41 \times 641\\
1{,}554{,}610{,}134 \times 1{,}554{,}610{,}135 \div 11\# &= 12{,}050{,}293 \;=\; 197 \times 61{,}169\\
1{,}626{,}111{,}200 \times 1{,}626{,}111{,}201 \div 11\# &= 13{,}184{,}240 \;=\; 2^4 \times 5 \times 97 \times 1{,}699\\
1{,}667{,}164{,}499 \times 1{,}667{,}164{,}500 \div 11\# &= 13{,}858{,}350 \;=\; 2 \times 3 \times 5^2 \times 11 \times 37 \times 227\\
1{,}681{,}048{,}655 \times 1{,}681{,}048{,}656 \div 11\# &= 14{,}090{,}136 \;=\; 2^3 \times 3 \times 251 \times 2{,}339\\
1{,}693{,}715{,}100 \times 1{,}693{,}715{,}101 \div 11\# &= 14{,}303{,}270 \;=\; 2 \times 5 \times 227 \times 6{,}301\\
1{,}897{,}433{,}460 \times 1{,}897{,}433{,}461 \div 11\# &= 17{,}950{,}962 \;=\; 2 \times 3 \times 107 \times 27{,}961\\
2{,}046{,}340{,}659 \times 2{,}046{,}340{,}660 \div 11\# &= 20{,}879{,}038 \;=\; 2 \times 59^2 \times 2{,}999\\
2{,}111{,}163{,}054 \times 2{,}111{,}163{,}055 \div 11\# &= 22{,}222{,}769\\
2{,}117{,}841{,}725 \times 2{,}117{,}841{,}726 \div 11\# &= 22{,}363{,}595 \;=\; 5 \times 67 \times 241 \times 277\\
2{,}185{,}445{,}625 \times 2{,}185{,}445{,}626 \div 11\# &= 23{,}814{,}125 \;=\; 5^3 \times 19 \times 37 \times 271\\
2{,}269{,}930{,}949 \times 2{,}269{,}930{,}950 \div 11\# &= 25{,}690{,}935 \;=\; 3 \times 5 \times 97 \times 17{,}657
\end{aligned}
$$

The lowest top prime found in the multiplier was 73, which is much larger than 31, so no π-complete products.

E. Computer Code

The Python program, Factorial_Search.py was created to search for solutions to the ABC Factorial Equation, and to gather statistics on the nature of the prime factorizations of the values attempted. It has been run up to a limit of $c = 10{,}000{,}000$ on a Mac Mini with Python version 3.12, and the SymPy module installed. Getting to 10,000,000 took about 3 weeks of runtime.

Default parameters cause the program to run to a limit of $c = 2{,}500$ and generate a statistics .csv file that records the values of internal counters during the run. Extended runs can be performed by using the "–limit n" switch to run the value of c to limit n. For example:

python Factorial_Search.py –limit 1000000 –no-stats

will run to $c = 1{,}000{,}000$ but not create a .csv file of the statistics. The "–progress-bar" switch will cause the program to display a progress bar while it runs, and the "–verbose" switch will cause the program to print π-complete quotients as it finds them.

Listing 1: Factorial Search Program Implementation

```


"""
Factorial Product Search Program - Revision 4.0
==============================================
This program searches for solutions to the equation a! x b! = c!
    where 1 < a < b < c-1.
Uses prime factor exponent lists and sliding window technique to
    manage memory usage.
Collects statistics to demonstrate end of possibility of solutions
    .

Example: 6! = 720 = 2^4 x 3^2 x 5^1 is represented as [4, 2, 1]
- 4 is the exponent of the first prime (2)
- 2 is the exponent of the second prime (3)
- 1 is the exponent of the third prime (5)

The only known non-trivial solution is 6! x 7! = 10!
It doesn't search b at c-1 because the equation collapses to a! =
    c in that case, and gives
the trivial solutions at every value of c that is a factorial
    number.

Run with: "python3 Factorial_Search.py" for a default run with
    statistics.
If run with default setting this program will search to 2,500! and
     generate a statistics file.

For long search use: python3 Factorial_Search.py --limit 1000000
    --no-stats
or any other large number for 1000000.

Run with --progrss-bar to supress periodic reporting.

Run with --verbose to watch search operation.

Author: Ken Clements
Date: Nov. 1, 2024
"""
import os
from tqdm import tqdm
import math
from  sympy import factorint, primepi
import csv
from datetime import datetime
import argparse



# Default Search Parameters

SEARCH_LIMIT =  2500 # Upper limit for factorial search
FIRST_C_TO_PROCESS = 7
WINDOW_SIZE = 10000  # Size of sliding window for factorial lists
WINDOW_OVERLAP = 500  # Number of entries to overlap between
```

```
        windows
    MAX_B_LOOKBACK = 490  # How far back to look for b values from c
                            # without flagging an error

    # Progress Reporting
    FACTOR_PROGRESS_INTERVAL = 10000  # How often to report factorial
        generation progress

    # Statistics Collection
    STATS_FILE_PREFIX = "factorial_search_stats"  # Prefix for stats
        file names

    class SearchStats:
        def __init__(self):
            timestamp = datetime.now().strftime('%Y%m%d_%H%M%S')
            self.filename = f"{STATS_FILE_PREFIX}_{timestamp}.csv"
            self.current_file = None
            self.headers = ['c', 'c_list_length',
                           'last_exp_too_large', 'b_attempts',
                           'a_attempts']

            # Create file and write headers
            with open(self.filename, 'w', newline='') as f:
                writer = csv.writer(f)
                writer.writerow(self.headers)

        def open_stats_file(self):
            """Open the stats file in append mode"""
            if self.current_file is None:
                self.current_file = open(self.filename, 'a', newline='
                    ')
                self.writer = csv.writer(self.current_file)

        def close_stats_file(self):
            """Close the stats file if it's open"""
            if self.current_file is not None:
                self.current_file.close()
                self.current_file = None
                self.writer = None

        def record_c_stats(self, c, stats):
            """Record statistics for a single c value"""
            if self.current_file is None:
                self.open_stats_file()

            self.writer.writerow([
                c, stats['c_list_length'], stats['length_mismatches'],
                stats['b_attempts'], stats['a_attempts']
            ])

        def __del__(self):
            """Ensure file is closed when object is destroyed"""
            self.close_stats_file()

    # END OF CLASS SearchStats

```

```python
class FactorialSearch:
    def __init__(self, limit: int = SEARCH_LIMIT, use_progress_bar: bool = False, verbose: bool = False, stats_enabled: bool = True):
        self.limit = limit
        self.factorial_prime_exponents_list = [[0], [0], [1]]  # Start with 2! at index 2
        self.a_factorial_prime_exponents_list = [[0], [0], [1]]  # Seperate buffer for compare with a
        self.prime_cache = [2]  # Initialize with first prime
        self.window_start = 0  # Offset for indexing
        self.stats = SearchStats() if stats_enabled else None
        self.use_progress_bar = use_progress_bar
        self.verbose = verbose
        self.c_for_a_attempts = []
        self.c_minus_b_candidate = []
        self.c_factorial_prime_exponents_for_a_attempts = []
        self.max_k = 0              # How deep into b have we gone?
        self.c_at_max_k = 0



    def compute_prime_exponent_list(self, n: int) -> list:
        """Get the prime factor exponent list for n using SymPy."""
        if n < 2:
            return []

        # Get prime factorization as dictionary
        factors = factorint(n)

        # Find the largest prime factor
        max_prime = max(factors.keys())

        # Get the index of the largest prime (how many primes up to max_prime)
        max_prime_idx = primepi(max_prime)

        # Create result list with zeros
        result = [0] * max_prime_idx

        # Fill in the exponents
        for prime, exp in factors.items():
            prime_idx = primepi(prime) - 1  # Convert prime to index (0-based)
            result[prime_idx] = exp

        return result



    def get_nth_prime(self, nth: int) -> int:
        """Get the nth prime number (0-based indexing)."""
        if nth < len(self.prime_cache):
            return self.prime_cache[nth]

        size_factor = 2
        s = (nth * size_factor)

        def get_primes(limit):
            sieve = [True] * limit
            for i in range(2, int(limit ** 0.5) + 1):
                if sieve[i]:
                    for j in range(i*i, limit, i):
                        sieve[j] = False
            return sieve

        while True:
            primes = get_primes(s)
            new_primes = [i for i in range(2, s) if primes[i]]
            if len(new_primes) > nth:
                self.prime_cache.extend(new_primes[len(self.
                    prime_cache):])
                return self.prime_cache[nth]
            size_factor += 1
            s = (nth * size_factor)


    def get_a_prime_exponent_list(self, idx):
        """ Get factorial from the special "a" buffer
            If not available, build up buffer until it is.
        """
        while idx >= len(self.a_factorial_prime_exponents_list):
            end_idx = len(self.a_factorial_prime_exponents_list) #
               Need to build
            f_list = self.compute_prime_exponent_list(end_idx)
            last_f_length = len(self.
                a_factorial_prime_exponents_list[-1])
            while len(f_list) < last_f_length:
                f_list.append(0) # make f_list the same size as
                    last buffer
            for j in range(last_f_length):
                f_list[j] += self.a_factorial_prime_exponents_list
                    [-1][j]
            self.a_factorial_prime_exponents_list.append(f_list)
            if self.verbose:
                print(f"Extending a_factorial_prime_exponents_list
                     to {idx}")

        return self.a_factorial_prime_exponents_list[idx]


    def get_prime_exponent_list(self, idx):
        """Get factorial factors adjusting for window offset"""
        return self.factorial_prime_exponents_list[idx - self.
            window_start]


    def slide_window(self, new_start, new_end):  # Add quiet
        parameter
        """Slide the window of factorial prime factors"""
        keep_start = max(2, new_start - WINDOW_OVERLAP)
        if keep_start > self.window_start:
            remove_count = keep_start - self.window_start
            self.factorial_prime_exponents_list = self.factorial_prime_exponents_list[remove_count:]
            self.window_start = keep_start

        current_end = self.window_start + len(self.factorial_prime_exponents_list)

        # Set up progress tracking for factorial generation
        remaining_factors = range(current_end, new_end + 1)
        if self.use_progress_bar:
           progress_iter = tqdm(remaining_factors,
                           desc="Generating factorial prime factors",
                           unit="factorial")
        else:
            progress_iter = remaining_factors

        for k in progress_iter:
            k_prime_exponents = self.compute_prime_exponent_list(k)
            last_f_length = len(self.factorial_prime_exponents_list[-1])
            while len(k_prime_exponents) < last_f_length:
                k_prime_exponents.append(0) # make f_list the same size as last buffer
            for j in range(last_f_length):
                k_prime_exponents[j] += self.factorial_prime_exponents_list[-1][j]
            self.factorial_prime_exponents_list.append(k_prime_exponents)

            if not self.use_progress_bar and k % FACTOR_PROGRESS_INTERVAL == 0:
                # Find the index of the last non-zero exponent
                print(f"Found {len(k_prime_exponents):,} prime factors up to {k:,}! max_k={self.max_k} at c = {self.c_at_max_k:,}")


    def search(self):
        """Main search function with sliding window"""
        window_size = WINDOW_SIZE
        current_start = 2
        total_exceptions = 0
        exceptions_past_10 = 0

        while current_start < self.limit:
            current_end = min(current_start + window_size, self.limit)
            print(f"\nProcessing window {current_start-1:,}! to {current_end:,}!")

            self.slide_window(current_start, current_end)

            if self.stats:
                self.stats.open_stats_file()

            exceptions = search_section(self, current_start,
                current_end)

            if exceptions:
                for a, b, c in exceptions:
                    print(f"\nFound exception: {a}! x {b}! = {c}!"
                        )
                    total_exceptions += 1
                    if c > 10:
                        exceptions_past_10 += 1

            if self.stats:
                self.stats.close_stats_file()

            if current_end >= self.limit:
                break

            current_start = current_end - WINDOW_OVERLAP

        print("\nSearch Complete!")
        print(f"Searched factorial products up to {self.limit
            -1:,}!")
        print(f"Total exceptions found: {total_exceptions}")
        print(f"Exceptions found past 10!: {exceptions_past_10}")
        if self.verbose:
            print(f"All c values at attempts to match a! were {
                self.c_for_a_attempts}")


# END OF CLASS FactorialSearch



"""
This search_section does the main work. It runs an outer loop
    searching c values
going up, a middle loop searching b values starting from c-2 and
    going down to
c/2, and an inner loop checking "a" going up from 2 to b for
    equality to the
subtractive difference (quotient) between the prime factor
    exponents of c and b.
All this depends on the equation a! x b! = c! were 1 < a < b < c-1
     having the
properties that the exponent list for c!/b! must not have the
    highest prime from c!
and must not have zeros before its highest prime factor.

"""

def search_section(searcher, start_idx, end_idx):
    exceptions = []
```

```

        # Set up progress tracking for search phase
        if searcher.use_progress_bar:
            c_range = tqdm(range(start_idx, end_idx),
                          desc="Searching for solutions",
                          unit="c")
        else:
            c_range = range(start_idx, end_idx)

        for c_idx in c_range:

            # Initialize statistics for this c value
            stats_for_c = {
                'length_mismatches': 0, # Count of b values rejected due to length
                'b_attempts': 0,        # Count of b values that passed length check
                'a_attempts': 0,        # Number of times it trys to find a!
            }

            c_factorial_prime_exponents = searcher.get_prime_exponent_list(c_idx)
            c_length = len(c_factorial_prime_exponents)      # How long is this factorization
            stats_for_c['c_list_length'] = c_length


            if not searcher.use_progress_bar and (c_idx + 1 == end_idx):
                print(f"Finished check for {c_idx:,}! with {c_length:,} prime factors")

            # Apply Bertrand-Chebyshev theorem - never search below c/2
            first_b = c_idx - 2
            last_b = max(c_idx // 2, 2)

            for b_idx in range(first_b, last_b, -1):
                b_factorial_prime_exponents = searcher.get_prime_exponent_list(b_idx)
                stats_for_c['b_attempts'] += 1
                if searcher.max_k < c_idx - b_idx:
                    searcher.max_k = c_idx - b_idx
                    searcher.c_at_max_k = c_idx

                assert b_idx > last_b - 1, "The b index should never get as low as c/2."


                if len(b_factorial_prime_exponents) != c_length: # Lengths must match
                    stats_for_c['length_mismatches'] += 1
                    break # Go get next c_idx


                # With matching length b_factorial_prime_exponents, we
                     need to find the highest
                # order prime exponents that don't subtract to zero
                # and check that subtraction equals 1
                d_idx = c_length-2
                for j in range(c_length-2, -1, -1):
                    d_idx = j # Find the high prime index for quotient
                    if c_factorial_prime_exponents[j] >
                        b_factorial_prime_exponents[j]:
                        break



                quotient_prime_exponents = [] # Start building the
                    difference in prime factor exponetns
                # Do the 2 power first
                quotient_prime_exponents.append(
                    c_factorial_prime_exponents[0] -
                    b_factorial_prime_exponents[0])
                if quotient_prime_exponents[0] == 0: # Cannot be zero
                    continue # move on to next b_idx
                error_flag = False

                for j in range(1,d_idx+1): # Start at power of 3
                    # Construct exponent list of c!/b! by subtracting
                        b! exponensts from c! exponents.
                    quotient_prime_exponents.append(
                        c_factorial_prime_exponents[j] -
                        b_factorial_prime_exponents[j])

                    if quotient_prime_exponents[j] == 0: # Cannot be
                        zero
                        error_flag = True
                        break # monve on to the next b_idx

                if error_flag:
                    continue # Move on to next b_idx


                # At this point there is an exponent list in quotient
                # that at least has no internal zero exponents, so
                    could match a factorial

                # Search for matching a!
                searcher.c_for_a_attempts.append(c_idx)
                searcher.c_minus_b_candidate.append(c_idx - b_idx)
                searcher.c_factorial_prime_exponents_for_a_attempts.
                    append(c_factorial_prime_exponents)

                if searcher.verbose:
                    print(f"--> Attempt to match a! at c = {c_idx:,}
                        with k = {c_idx - b_idx}")

                stats_for_c['a_attempts'] += 1  # Count the attempt

                # Search upward from 2 for matching "a"
```

```
            error_flag = False
            for a_idx in range(2, b_idx):

                a_factors = searcher.get_a_prime_exponent_list(
                    a_idx)

                if len(a_factors) > len(quotient_prime_exponents):
                    break  # All higher a values will be too large

                if len(a_factors) < len(quotient_prime_exponents):
                    continue  # Try next a value

                # Check each exponent in order
                for i in range(len(quotient_prime_exponents)):
                    if a_factors[i] < quotient_prime_exponents[i]:
                        error_flag = True
                        break
                    if a_factors[i] > quotient_prime_exponents[i]:
                        break
                else:  # All exponents match
                    exceptions.append((a_idx, b_idx, c_idx))
                    break

            if error_flag:
                break

        # Record stats for this c value if there was any activity
        if searcher.stats and (stats_for_c['b_attempts'] > 0 or
                             stats_for_c['length_mismatches'] > 0)
                                 :
            searcher.stats.record_c_stats(c_idx, stats_for_c)

    return exceptions

def parse_arguments():
    parser = argparse.ArgumentParser(
        description='Factorial Product Search Program - Searches
            for solutions to a! x b! = c!',
        formatter_class=argparse.ArgumentDefaultsHelpFormatter
    )

    parser.add_argument('--limit', type=int, default=SEARCH_LIMIT,
                       help='Upper limit for factorial search')
    parser.add_argument('--window-size', type=int, default=
        WINDOW_SIZE,
                       help='Size of sliding window for factorial
                           lists')
    parser.add_argument('--no-stats', action='store_true',
                       help='Disable statistics collection')
    parser.add_argument('--progress-bar', action='store_true',
                       help='Use progress bar instead of interval
                           reporting')
    parser.add_argument('--verbose', action='store_true',
                       help='Print out prime factorization for
                           candidate c!/b!')


    return parser.parse_args()

if __name__ == "__main__":
    #  Clear the screen
    os.system('cls' if os.name == 'nt' else 'clear')
    args = parse_arguments()

    searcher = FactorialSearch(limit=args.limit+1,
        use_progress_bar=args.progress_bar, \
                    verbose=args.verbose, stats_enabled=not args.
                        no_stats)
    searcher.search()


# END OF PROGRAM
```

F. π-complete Analyzer Program

This Python program, Pi-complete_Analyzer.py, examines the prime factorizations of products of k-tuples to find cases in which no prime factors are missing (that is, they have the π-complete property). It does this in three sections. In the first section it looks at the k-tuples for $k = 2$ through $k = 5$ with a high target for c. It prints the results found and then looks at the k-tuples for $k = 6$ through $k = 25$ with the target for c at 1,000, and prints the results. Finally, it sets the target for c to 1,200 and records the highest π-complete values it finds for $k = 6$ through $k = 500$ and prints the results.

The first section examines tuple products by looking at the combined prime factorizations of their successive integer factors. Thus, it moves through those products at a square rate for pairs, a cubic rate for 3-tuples, etc.

The program has been run on a Mac Mini with Python 3.12 and the SymPy module installed. On that machine, it takes about two hours to run when the first part is going up to $c = 10{,}000{,}000$. The output of the last run is listed below after the code.

As a further test to find possible consecutive integer pairs with products up to 10,000,000,000,000 the program was modified to generate π-complete candidate products, $r\#m$, by primorial order for orders $5\# = 2{,}310$ through $11\# = 200{,}560{,}490{,}130$ but no further 2-tuple products were found. Whereas the program had generated integer pairs and checked the product for π-completeness, the modified search provided corroboration by generating the product candidate by $r\#m$ (m being successive numbers that were found to be p_r-smooth) and checking that for a pair of consecutive integer factors. This check was done by multiplying the floor and ceiling integers of the square root to test if that product matched the generated $r\#m$. The square root was checked before the floor and ceiling functions were called to be sure it was within valid numeric resolution range.

The limits for this $r\#m$ corroborating search for each $r\#$ order were:

$$\begin{aligned}
5\#4{,}327{,}178{,}240 &= 9{,}995{,}781{,}734{,}400\\
6\#332{,}972{,}640 &= 9{,}999{,}168{,}379{,}200\\
7\#19{,}584{,}000 &= 9{,}997{,}827{,}840{,}000\\
8\#1{,}030{,}029 &= 9{,}990{,}961{,}991{,}010\\
9\#44{,}800 &= 9{,}994{,}560{,}576{,}000\\
10\#1{,}540 &= 9{,}963{,}327{,}574{,}200\\
11\#49 &= 9{,}827{,}464{,}016{,}370
\end{aligned}$$

(For bevity, the listing of this modified Pi-complete Analyzer corroboration test program is not included in the listing below.)

Listing 2: Pi-complete Analyzer Program Implementation

```

    """ This program searches the products of k-tuples for for results with
    no zero exponents in their prime factorizatoins. These Pi-complete numbers are found
    for values of k starting at 6 and going to MAX_K. The k-tuple products are from descending
    consecutive integers c(c-1)(c-2)...(c-k+1) where c starts at the value
    k+1. The search goes up to c = C_LIMIT. The default values are MAX_K = 500 and C_LIMIT = 1{,}200.
    At the end of the search, the program prints out a list of the c value that produced
    the last pi-complete product for k.

    Written by Ken Clements Nov. 22, 2024

"""
import os
import math
from sympy import factorial, factorint, primepi

LOW_K_SEARCH_LIMIT = 50_000_000 # Takes a few days to hit this on a Mac Mini
HIGH_K_SEARCH_LIMIT = 1200      # Max c to check on deep k-tuples
HIGH_K_DEPTH = 500              # Max k-tuple depth to check

def get_prime_factor_exponent_list(n: int) -> list:
    """Get the prime factor exponent list for n using SymPy."""
    if n < 2:
        return []

    # Get prime factorization as dictionary
    factors = factorint(n)

    # Find the largest prime factor
    max_prime = max(factors.keys())

    # Get the index of the largest prime (how many primes up to max_prime)
    max_prime_idx = primepi(max_prime)

    # Create result list with zeros
    result = [0] * max_prime_idx

    # Fill in the exponents
    for prime, exp in factors.items():
        prime_idx = primepi(prime) - 1  # Convert prime to index (0-based)
        result[prime_idx] = exp

    return result

""" ---------------------------------------------------------------------
```

```
     """


def is_mono(product_pfel):
    """ Check if monotonic non-decreasing """
    for idx in range(len(product_pfel)-2, -1, -1):
        if product_pfel[idx] < product_pfel[idx+1]:
            return False
    return True

"""
    ---------------------------------------------------------------------
     """


def has_FEF(product_pfel):
    """ Check product for Factorial Exponent Form """
    if product_pfel == []:
        return False

    if product_pfel[-1] != 1:
        return False
    return is_mono(product_pfel)

"""
    ---------------------------------------------------------------------
     """


def print_results(no_fail, k, print_limit):
    """ Print the pi-complete values for c found by searches. Put the
         values
        that are also FEF in ()
    """
    print_count = 0
    for idx in range(len(no_fail)):
        n_test = no_fail[idx]
        product = n_test
        for times_m1 in range(k):
            product *= n_test - (times_m1+1)
        product_pfel = get_prime_factor_exponent_list(product)
        if has_FEF(product_pfel):
            print(f"({n_test})", end="") # It's a perfect FEF number
        else:
            print(f"{n_test}", end="") # If not FEF, just print number
        print(",", end="")
        print_count += 1
        if print_count > print_limit:
            print("\b ")
            print_count = 0
    print("\b ")

"""
    ---------------------------------------------------------------------
     """

```

```python


def process_loop(depth, search_start, search_end) -> list:
    """ This loop does the main work. It uses a FIFO buffer of prime factorizations
        to look for missing prime factors across k-tuples. Depth, is how deep to
        make the FIFO, i.e. how many factors in the tuple.
    """
    progress_reporting_block = search_end // 10 # Report every 10%
    cm_pl = [[]]      # FIFO for lists of factor prime factorizations
    cm_no_fail = [[]]    # Lists of pi-complete reaults

    for idx in range(depth):     # Initialize buffers
        cm_pl.append(get_prime_factor_exponent_list(search_start-(idx+1)))
        cm_no_fail.append([])

    for c in range(search_start, search_end+1):

        for idx in range(depth-1, 0, -1):
            cm_pl[idx] = cm_pl[idx-1]
        cm_pl[0] = get_prime_factor_exponent_list(c)

        for idx in range(depth-1):
         # Check for zeros in combined factorization
            zero_fail = False

            max_length = 0
            for m_cycles in range(idx+2):
                max_length = max(max_length, len(cm_pl[m_cycles]))


            for j in range(max_length - 1):
                power_sum = 0
                for m_cycles in range(idx+2):
                    power_sum += cm_pl[m_cycles][j] if j<len(cm_pl[m_cycles]) else 0
                if power_sum == 0:
                    zero_fail = True
                    break
            if not zero_fail and c > idx + 2:
                cm_no_fail[idx].append(c)

        if c % progress_reporting_block == 0:
            print(f"  {c:,} Numbers checked.", end="\r")
    print("")

    return cm_no_fail    # Return list of found pi-complete values for c

""" --------------------------------------------------------------------- """


```

```python

def main():

    #  Clear the screen
    os.system('cls' if os.name == 'nt' else 'clear')

    # Calculate and display pi-complete sequences for k = 2..5
    max_k = 5 # Only going up to 5-tuple
    search_start = max_k + 1    # Can't start lower than tuple number +1
    search_end = LOW_K_SEARCH_LIMIT  # Value for how high to search c
    block_length = max_k
    print_limit = 10

    print(f"\n\nThis program searches for consecutive k-tuples whose product,")
    print(f"in the form of c(c-1)...(c-k+1) for k starting at 2,")
    print(f"has no internal zeros in its prime factorization exponents.\n")
    print(f"The search starts from c = {search_start} and ends at {search_end:,}.")
    print("----------------------------------------------------------------")

    pi-complete_list = process_loop(block_length, search_start, search_end)

    print(f"Search concluded at c = {search_end:,}")
    print("\nShowing pi-complete k-tuple start values.")
    print("If the number is in '()' then the product has the FEF property.")
    print("----------------------------------------------------------------")

    for idx in range(block_length-1):
        if idx == 0:
            print("\nc(c-1) non-zeros are where c = ", end=" ")
            print_results(pi-complete_list[0], 1, print_limit)
        elif idx == 1:
            print("\nc(c-1)(c-2) non-zeros are where c = ", end=" ")
            print_results(pi-complete_list[idx], idx, print_limit)
        else:
            print(f"\nc(c-1)...(c-{idx+1}) non-zeros are where c=", end=" ")
            print_results(pi-complete_list[idx], idx, print_limit)

"""---------------------------------------------------------------------"""


    # Now, calculate and display the pi-complete sequences for large k
    search_start = 6
    block_length = 25
    search_end = HIGH_K_SEARCH_LIMIT     # Lower end for large k

    print(f"\n\nNext, the program searches for larger k values.")
    print(f"The search starts from {search_start} and ends at {
        search_end:,}.")
    print("
        ----------------------------------------------------------------
        ")

    pi-complete_list = process_loop(block_length, search_start,
        search_end)

    print(f"Search concluded at c = {search_end:,}")
    print("\nShowing pi-complete k-tuple start values.")
    print("If the number is in '()' then the product as FEF property."
        )
    print("
        ----------------------------------------------------------------
        ")


    for idx in range(search_start-2, block_length-1):
        if idx == 0:
            print("\nc(c-1) non-zeros are where c=", end=" ")
            print_results(pi-complete_list[0], 1, print_limit)
        else:
            print(f"\nc(c-1)...(c-{idx+1}) non-zeros are where c=",
                end=" ")
            print_results(pi-complete_list[idx], idx, print_limit)


"""
    ---------------------------------------------------------------------
     """

    # Calculate and display the list of pi-complete limits for large k
    search_start = 6
    search_end = HIGH_K_SEARCH_LIMIT
    block_length = HIGH_K_DEPTH
    print_limit = 20

    print(f"\n\nFinally, the program searches deep consecutive k-
        tuples")
    print(f"and prints the list of max c values for k range 6 to {
        HIGH_K_DEPTH}.")
    print(f"The search starts from c = {search_start} and ends at c =
        {search_end:,}.")
    print("
        ----------------------------------------------------------------
        ")

    pi-complete_list = process_loop(block_length, search_start,
        search_end)

    print(f"Search concluded at c = {search_end:,}")
    print("\nShowing pi-complete k-tuple limits for c starting at k =
        6.")
    print("
        -----------------------------------------------------------------
        ")

    print_count = 0
    for idx in range(search_start-2, block_length-1):

        print(f"{pi-complete_list[idx][-1]:3}", end=", ")
        print_count += 1
        if print_count == print_limit:
            print("\b\b  ")
            print_count = 0
    if print_count > 0:
        print("\b\b  ")

    print("")
    print("Results Concluded")
    print("END OF PROGRAM")

"""
    -----------------------------------------------------------------------
     """


if __name__ == "__main__":
    main()

Program output:

    This program searches for consecutive k-tuples whose product,
    in the form of c(c-1)...(c-k+1) for k starting at 2,
    has no internal zeros in its prime factorization exponents.

    The search starts from c = 6 and ends at 50,000,000.
    ------------------------------------------------------------------------

      50,000,000 Numbers checked.
    Search concluded at c = 50,000,000

    Showing pi-complete k-tuple start values.
    If the number is in '()' then the product has the FEF property.
    ------------------------------------------------------------------------


    c(c-1) non-zeros are where c =  (6),9,10,(15),(16),(21),25,(36)
        ,(81),126,(225)
    (385),441,540,(715),1716,(2080),2401,3025,4375,9801,12376
    123201,194481,633556

    c(c-1)(c-2) non-zeros are where c =  (6),10,(16)
        ,22,50,56,100,352,442

    c(c-1)...(c-3) non-zeros are where c= (6),(7),(10),66

    c(c-1)...(c-4) non-zeros are where c= (6),(7),(8),(10),11,14,15
```

```


Next, the program searches for larger k values.
The search starts from 6 and ends at 1,000.
--------------------------------------------------------------------------

  1,000 Numbers checked.
Search concluded at c = 1,000

Showing pi-complete k-tuple start values.
If the number is in '()' then the product as FEF property.
-------------------------------------------------------------------------


c(c-1)...(c-5) non-zeros are where c= (7),(8),(9),(10),(11)
    ,12,(14),(15),16

c(c-1)...(c-6) non-zeros are where c= (8),(9),(10),(11),(12)
    ,13,(14),(15),(16),17

c(c-1)...(c-7) non-zeros are where c= (9),(10),(11),(12),(13),(14)
    ,(15),(16),(17),18

c(c-1)...(c-8) non-zeros are where c= (10),(11),(12),(13),14,(15)
    ,(16),(17),(18),19

c(c-1)...(c-9) non-zeros are where c= (11),(12),(13),14,(15),(16)
    ,(17),(18),(19),20,22
26

c(c-1)...(c-10) non-zeros are where c= (12),(13),(14),(15),(16)
    ,(17),(18),(19),(20),21,(22)
23,(26),27

c(c-1)...(c-11) non-zeros are where c= (13),(14),(15),(16),(17)
    ,(18),(19),(20),(21),(22),(23)
24,(26),(27),28

c(c-1)...(c-12) non-zeros are where c= (14),(15),(16),(17),(18)
    ,(19),(20),(21),(22),(23),(24)
25,(26),(27),(28),29

c(c-1)...(c-13) non-zeros are where c= (15),(16),(17),(18),(19)
    ,(20),(21),(22),(23),(24),(25)
(26),(27),(28),(29),30

c(c-1)...(c-14) non-zeros are where c= (16),(17),(18),(19),(20)
    ,(21),(22),(23),(24),(25),26
(27),(28),(29),(30),31

c(c-1)...(c-15) non-zeros are where c= (17),(18),(19),(20),(21)
    ,(22),(23),(24),(25),26,27
(28),(29),(30),(31),32,34,38

c(c-1)...(c-16) non-zeros are where c= (18),(19),(20),(21),(22)
    ,(23),(24),(25),(26),27,28
```

```
(29),(30),(31),(32),33,(34),35,(38),39

c(c-1)...(c-17) non-zeros are where c= (19),(20),(21),(22),(23)
    ,(24),(25),(26),(27),28,29
(30),(31),(32),(33),(34),(35),36,(38),39,40,46


c(c-1)...(c-18) non-zeros are where c= (20),(21),(22),(23),(24)
    ,(25),(26),(27),(28),29,30
(31),(32),(33),34,(35),(36),37,(38),(39),40,41
(46),47

c(c-1)...(c-19) non-zeros are where c= (21),(22),(23),(24),(25)
    ,(26),(27),(28),(29),30,31
(32),(33),34,35,(36),(37),(38),(39),(40),41,42
(46),(47),48

c(c-1)...(c-20) non-zeros are where c= (22),(23),(24),(25),(26)
    ,(27),(28),(29),(30),31,32
(33),34,35,36,(37),38,(39),(40),(41),42,43
(46),(47),(48),49

c(c-1)...(c-21) non-zeros are where c= (23),(24),(25),(26),(27)
    ,(28),(29),(30),(31),32,(33)
34,35,36,37,38,39,(40),(41),(42),43,44
(46),(47),(48),(49),50

c(c-1)...(c-22) non-zeros are where c= (24),(25),(26),(27),(28)
    ,(29),(30),(31),(32),(33),(34)
35,36,37,38,39,40,(41),(42),(43),(44),45
(46),(47),(48),49,(50),51

c(c-1)...(c-23) non-zeros are where c= (25),(26),(27),(28),(29)
    ,(30),(31),(32),(33),(34),(35)
36,37,38,(39),40,41,(42),(43),(44),(45),(46)
(47),(48),49,(50),51,52

c(c-1)...(c-24) non-zeros are where c= (26),(27),(28),(29),(30)
    ,(31),(32),(33),(34),(35),(36)
37,38,(39),(40),41,42,(43),(44),(45),46,(47)
(48),49,(50),51,(52),53

Finally, the program searches deep consecutive k-tuples
and prints the list of max c values for k range 6 to 500.
The search starts from c = 6 and ends at c = 1,200.
-------------------------------------------------------------------

  1,200 Numbers checked.
Search concluded at c = 1,200

Showing pi-complete k-tuple limits for c starting at k = 6.
------------------------------------------------------------------

 16,  17,  18,  19,  26,  27,  28,  29,  30,  31,  38,  39,  46,
    47,  48,  49,  50,  51,  52,  53
 62,  63,  64,  65,  66,  67,  68,  69,  74,  75,  76,  77,  78,
        79,  86,  87,  94,  95,  96,  97
   98,  99, 106, 107, 108, 109, 110, 111, 112, 113, 122, 123, 124,
        125, 126, 127, 128, 129, 134, 135
  136, 137, 146, 147, 148, 149, 150, 151, 152, 153, 158, 159, 166,
      167, 168, 169, 178, 179, 180, 181
  182, 183, 184, 185, 186, 187, 188, 189, 194, 195, 196, 197, 198,
      199, 226, 227, 228, 229, 230, 231
  232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244,
      245, 246, 247, 248, 249, 254, 255
  262, 263, 264, 265, 278, 279, 280, 281, 282, 283, 284, 285, 286,
      287, 288, 289, 290, 291, 292, 293
  302, 303, 304, 305, 306, 307, 314, 315, 316, 317, 318, 319, 320,
      321, 326, 327, 334, 335, 336, 337
  338, 339, 346, 347, 348, 349, 362, 363, 364, 365, 366, 367, 368,
      369, 370, 371, 372, 373, 374, 375
  376, 377, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408,
      409, 422, 423, 424, 425, 426, 427
  428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440,
      441, 446, 447, 448, 449, 450, 451
  458, 459, 466, 467, 468, 469, 482, 483, 484, 485, 486, 487, 488,
      489, 490, 491, 492, 493, 494, 495
  502, 503, 504, 505, 506, 507, 514, 515, 516, 517, 518, 519, 526,
      527, 528, 529, 530, 531, 532, 533
  542, 543, 544, 545, 546, 547, 548, 549, 566, 567, 568, 569, 570,
      571, 586, 587, 588, 589, 590, 591
  592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604,
      605, 606, 607, 608, 609, 634, 635
  636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648,
      649, 650, 651, 652, 653, 654, 655
  662, 663, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684,
      685, 686, 687, 688, 689, 690, 691
  698, 699, 706, 707, 708, 709, 718, 719, 720, 721, 722, 723, 724,
      725, 726, 727, 734, 735, 736, 737
  738, 739, 746, 747, 748, 749, 750, 751, 752, 753, 758, 759, 766,
      767, 768, 769, 778, 779, 780, 781
  782, 783, 784, 785, 786, 787, 788, 789, 802, 803, 804, 805, 806,
      807, 818, 819, 820, 821, 822, 823
  824, 825, 826, 827, 828, 829, 842, 843, 844, 845, 846, 847, 848,
      849, 850, 851, 852, 853, 854, 855
  856, 857, 866, 867, 868, 869, 870, 871, 872, 873, 878, 879, 886,
      887, 888, 889, 898, 899, 900, 901
  902, 903, 904, 905, 906, 907, 908, 909, 914, 915, 934, 935, 936,
      937, 938, 939, 940, 941, 942, 943
  944, 945, 946, 947, 948, 949, 958, 959, 960, 961, 962, 963, 964,
      965, 966, 967, 968, 969, 982, 983
  984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 998, 999, 1018,
      1019, 1020

  Results Concluded
  END OF PROGRAM
```