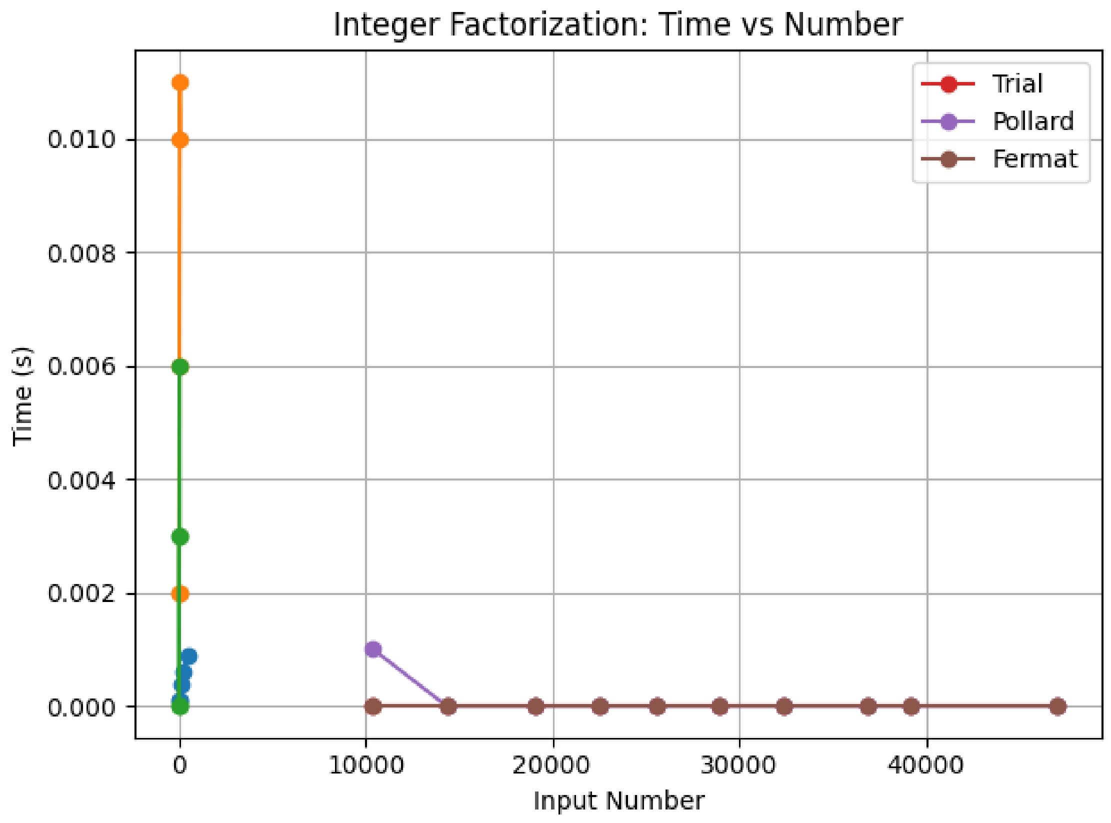
Practical Assignment No. 4 **Title:** Integer Factorization Problem

```
import math
import time
import random
import matplotlib.pyplot as plt

# 1. Trial Division
def trial_division(n):
    factors = []
    for i in range(2, int(math.isqrt(n)) + 1):
        while n % i == 0:
            factors.append(i)
            n //= i
    if n > 1:
        factors.append(n)
    return factors

# Helper: primality check
def is_prime(n):
    if n < 2:
        return False
    for i in range(2, int(math.isqrt(n)) + 1):
        if n % i == 0:
            return False
    return True

# 2. Pollard's Rho (Recursive full factorization)
def pollards_rho(n):
    if n == 1:
        return []
    if is_prime(n):
        return [n]
    if n % 2 == 0:
        return [2] + pollards_rho(n // 2)

    def rho(n):
        x = random.randrange(2, n)
        y = x
        c = random.randrange(1, n)
        d = 1
        while d == 1:
            x = (pow(x, 2, n) + c) % n
            y = (pow(y, 2, n) + c) % n
            y = (pow(y, 2, n) + c) % n
            d = math.gcd(abs(x - y), n)
            if d == n:
                return rho(n)
        return d
    factor = rho(n)
    return pollards_rho(factor) + pollards_rho(n // factor)

# 3. Fermat's Factorization (works well for numbers with close prime factors)
def fermat_factor(n):
    if n <= 0:
        raise ValueError("Input must be positive.")
    if n % 2 == 0:
        return [2, n // 2]
    a = math.isqrt(n)
    if a * a < n:
        a += 1
    b2 = a * a - n
    while math.isqrt(b2) ** 2 != b2:
        a += 1
        b2 = a * a - n
    b = math.isqrt(b2)
    return [a - b, a + b]

# Generate 10 big composite numbers (products of 2 primes)
numbers = [101 * 103, 113 * 127, 137 * 139, 149 * 151, 157 * 163,
           167 * 173, 179 * 181, 191 * 193, 197 * 199, 211 * 223]

results = {"Trial": [], "Pollard": [], "Fermat": []}

for n in numbers:
    # Trial Division
    start = time.time()
    trial_division(n)
    results["Trial"].append(time.time() - start)

    # Pollard's Rho
    start = time.time()
    pollards_rho(n)
    results["Pollard"].append(time.time() - start)

    # Fermat's Factor
    start = time.time()
    fermat_factor(n)
    results["Fermat"].append(time.time() - start)
```

```
# Plotting
for method in results:
    plt.plot(numbers, results[method], label=method, marker='o')

plt.xlabel("Input Number")
plt.ylabel("Time (s)")
plt.title("Integer Factorization: Time vs Number")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```


Integer Factorization: Time vs Number
Trial
Pollard
Fermat
Time (s)
Input Number