**Lab 6 (Parallel Programming)**

**NAME:** ADITYA MISHRA

**REG. NO:** RA2111003011817

1. Write a program that uses multiple threads to solve the problem of finding the integer in the range 1 to 10000 that has the largest number of divisors, but for the range 1 to 100000. At the end of the program, output the elapsed time, the integer that has the largest number of divisors, and the number of divisors that it has.

import time

import threading

def count\_divisors(n):

count = 0

for i in range(1, n+1):

if n % i == 0:

count += 1

return count

def find\_largest\_divisors(start, end, result):

max\_divisors = 0

max\_num = 0

for n in range(start, end+1):

divisors = count\_divisors(n)

if divisors > max\_divisors:

max\_divisors = divisors

max\_num = n

result.append((max\_num, max\_divisors))

if \_\_name\_\_ == '\_\_main\_\_':

num\_threads = 4

chunk\_size = 100000 // num\_threads

results = []

start\_time = time.time()

threads = []

for i in range(num\_threads):

start = i \* chunk\_size + 1

end = start + chunk\_size - 1

if i == num\_threads - 1:

end = 100000

thread = threading.Thread(target=find\_largest\_divisors, args=(start, end, results))

threads.append(thread)

thread.start()

for thread in threads:

thread.join()

max\_num, max\_divisors = max(results, key=lambda x: x[1])

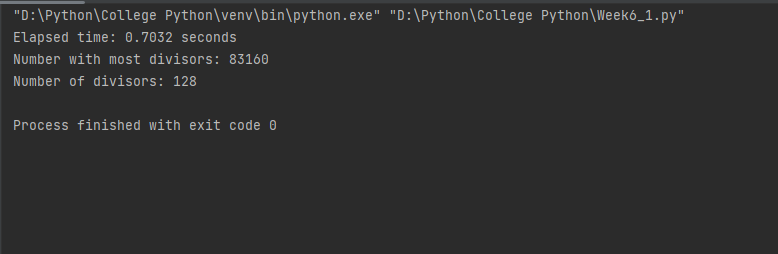
elapsed\_time = time.time() - start\_time

print(f"Elapsed time: {elapsed\_time:.3f} seconds")

print(f"Number with max divisors: {max\_num}")

print(f"Number of divisors: {max\_divisors}")o

**Output-**

****

5) Write a multi processing program in which the first process prints odd numbers between

100 to 200. Second Process print the prime number. Between 200 to 300. And third process

generate Armstrong number between 100 to 300.

import multiprocessing

def is\_prime(n):

if n <= 1:

return False

for i in range(2, int(n \*\* 0.5) + 1):

if n % i == 0:

return False

return True

def is\_armstrong(n):

digits = [int(x) for x in str(n)]

num\_digits = len(digits)

sum\_of\_powers = sum([x \*\* num\_digits for x in digits])

return sum\_of\_powers == n

def print\_odd\_numbers():

for i in range(101, 200, 2):

print(i)

def print\_prime\_numbers():

for i in range(200, 300):

if is\_prime(i):

print(i)

def print\_armstrong\_numbers():

for i in range(100, 300):

if is\_armstrong(i):

print(i)

if \_\_name\_\_ == '\_\_main\_\_':

p1 = multiprocessing.Process(target=print\_odd\_numbers)

p2 = multiprocessing.Process(target=print\_prime\_numbers)

p3 = multiprocessing.Process(target=print\_armstrong\_numbers)

p1.start()

p2.start()

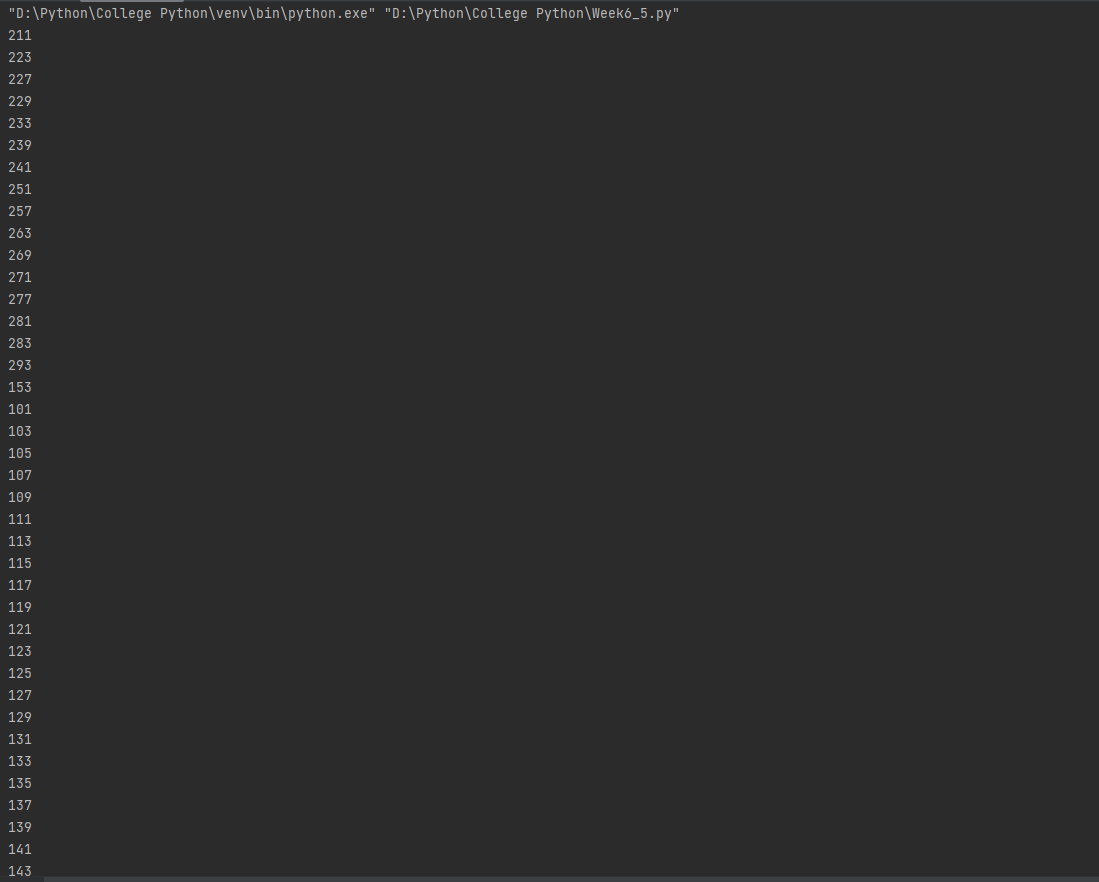
p3.start()

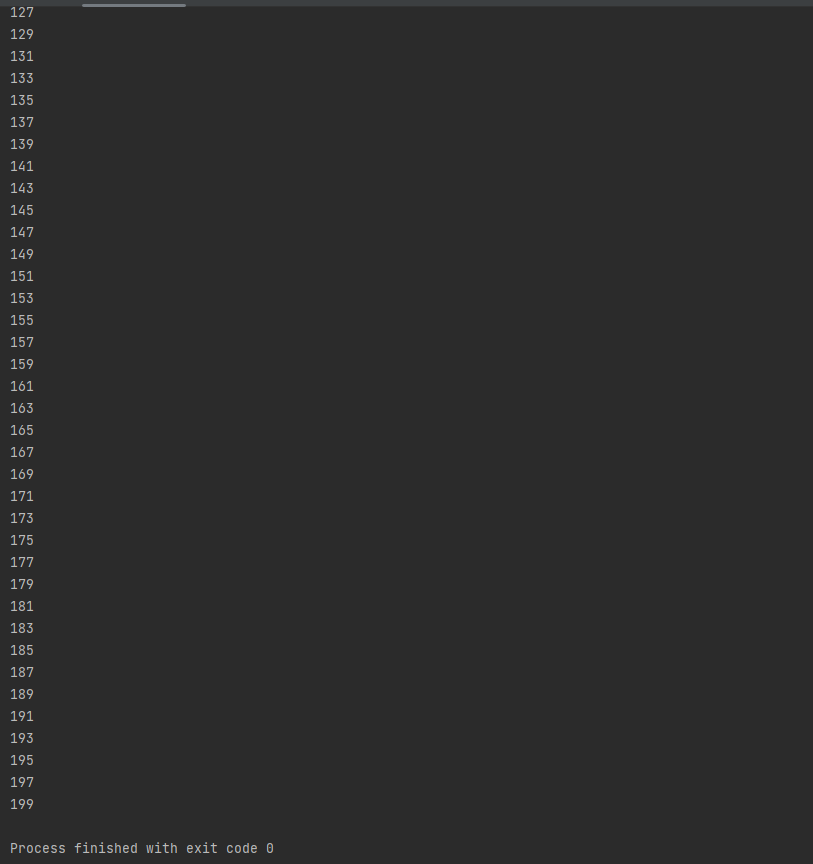
p1.join()

p2.join()

p3.join()

**OUTPUT:**

****

****