Data Analysis and Algorithm

Practical 9

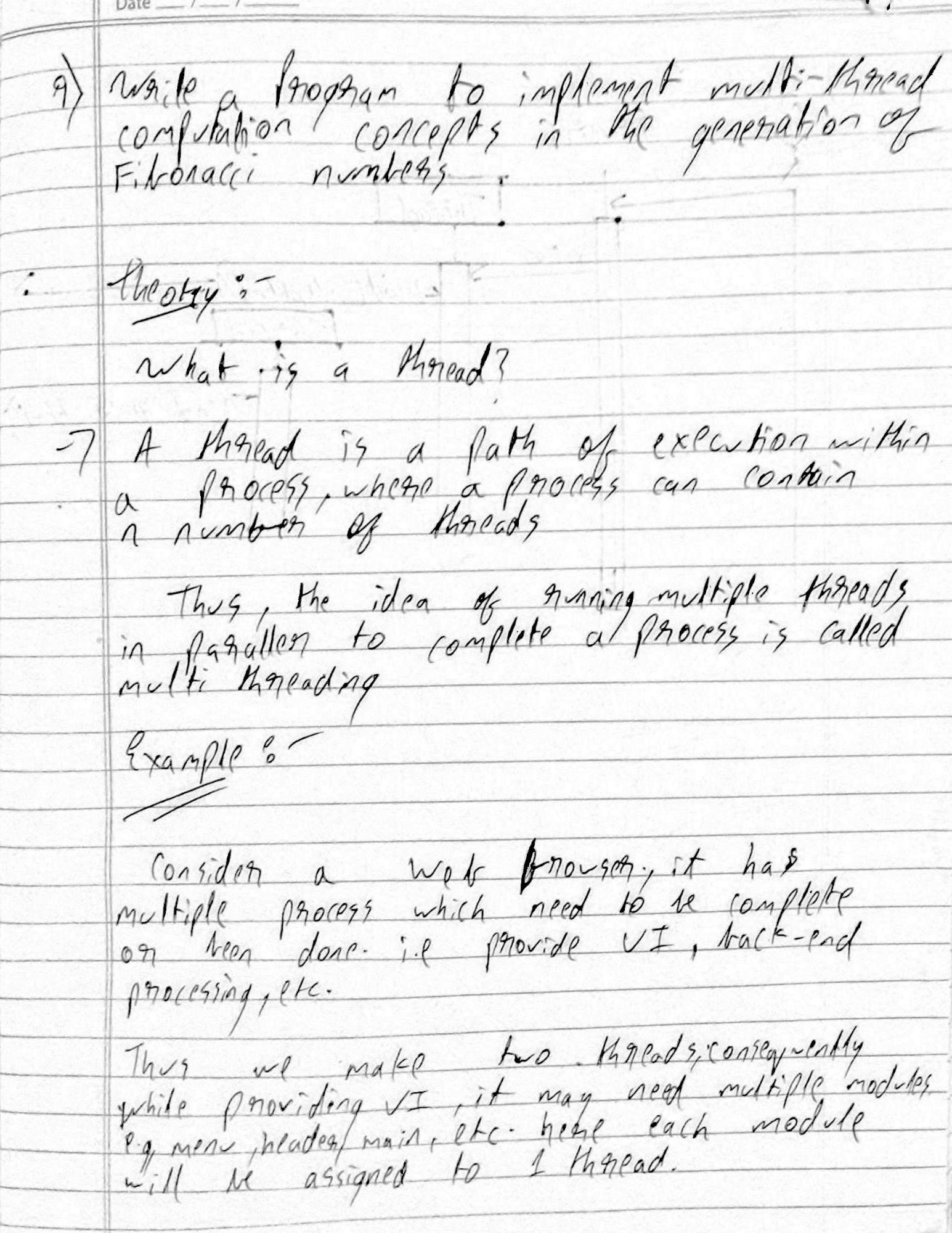
Write a program to implement multi threaded computation concepts in the generation of Fibonacci numbers.

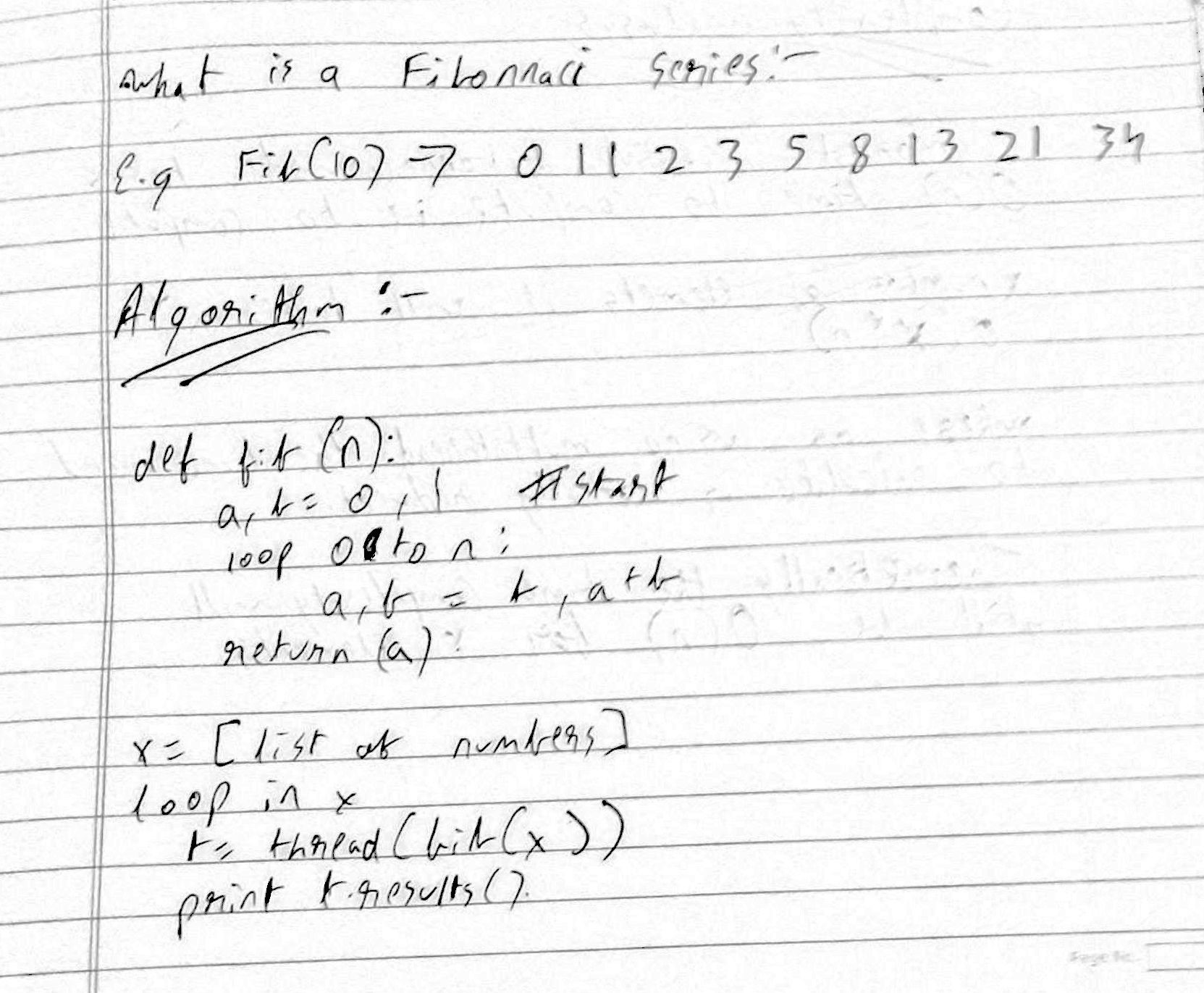
Date.: 19-10-21

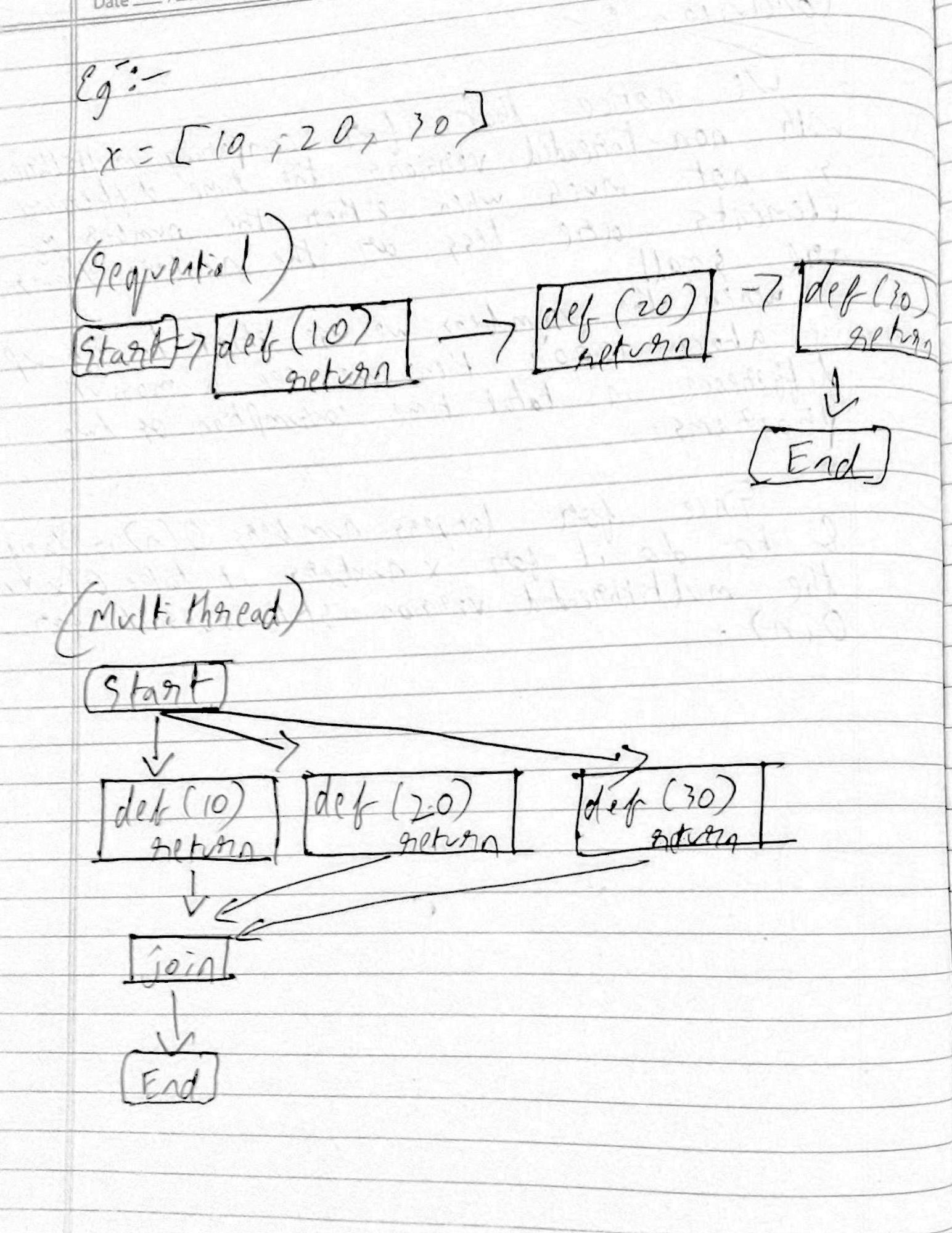
Name – Yash Vasudeo Prajapati

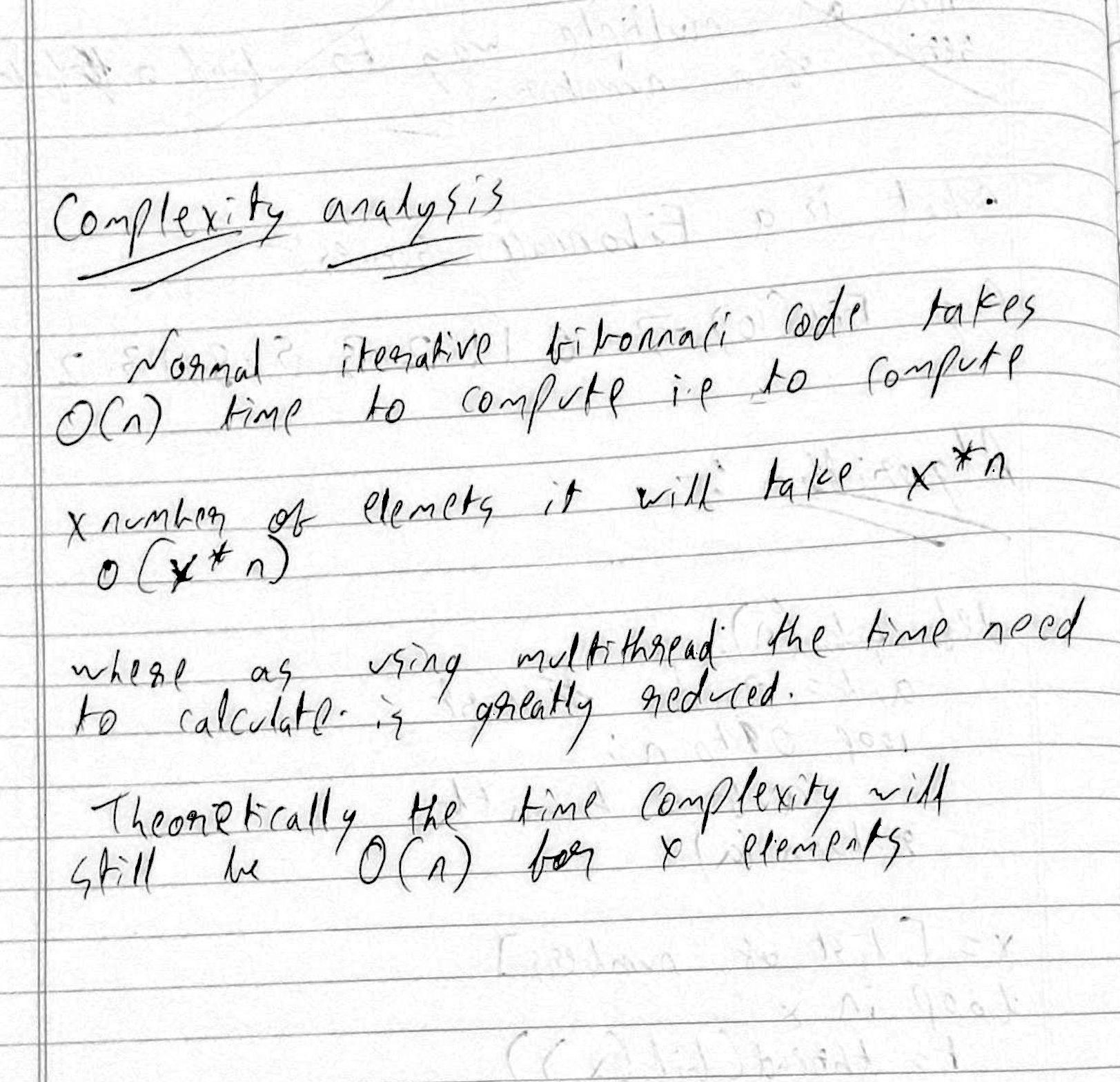
Rollno - 022

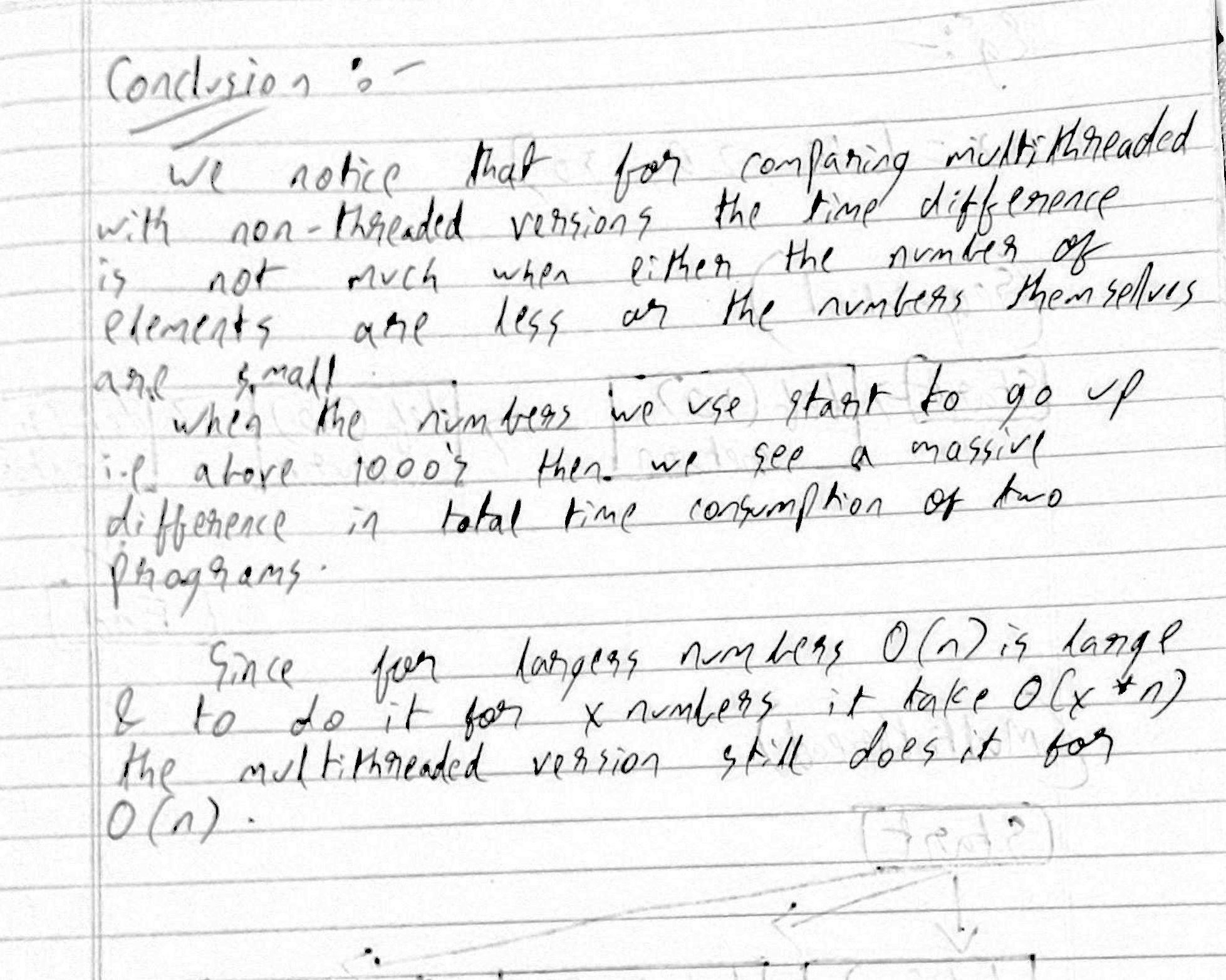
MSc. Computer Science











Program:-

Multi-Threaded

import queue, threading, random

from timeit import default\_timer as timer

#setting up random list

x=[]

n=200

for i in range(n):

x.append(random.randint(1000,10000))

print("With Threading in range (1000, 10000)")

#timer start

start = timer()

q = queue.Queue()

def fib(n):

a, b = 0, 1

for i in range(0, n):

a, b = b, a + b

q.put((n, a))

return

for i in x:

t = threading.Thread(target=fib, args = (i,))

t.daemon = True

t.start()

while not q.empty():

n, f = q.get()

#print ("{0}: {1}".format(n, f))

print("Time taken %s seconds" % (timer() - start))

Without Multi-Thread

import random

from timeit import default\_timer as timer

#setting up random list

x=[]

n=2000

for i in range(n):

x.append(random.randint(1000,10000))

print("Without Threading in range (1000, 10000)")

#timer start

start = timer()

arr = []

def fib(n):

a, b = 0, 1

for i in range(0, n):

a, b = b, a + b

return (a)

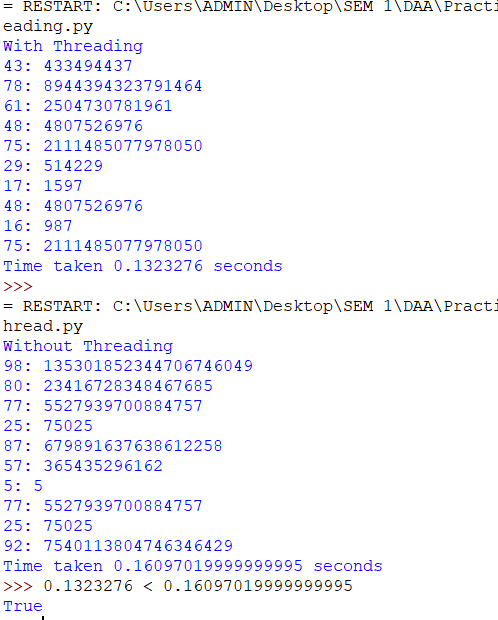
for i in range(0,len(x)):

arr.append(fib(x[i]))

#print("{0}: {1}".format(x[i], arr[i]))

print("Time taken %s seconds" % (timer() - start))

Output:-



Increasing the size of numbers

