Pg1

def fibonacci(n):

if n <= 1:

return n

else:

return fibonacci(n - 1) + fibonacci(n - 2)

num\_terms = int(input("Enter the number of terms: "))

if num\_terms <= 0:

print("Please enter a positive integer")

else:

print("Fibonacci sequence:")

for i in range(num\_terms):

print(fibonacci(i))

pg2

def add(x, y):

return x + y

def subtract(x, y):

return x - y

def multiply(x, y):

return x \* y

def divide(x, y):

if y == 0:

return "Error! Division by zero."

return x / y

while True:

print("\nSelect an operation:")

print("1. Addition")

print("2. Subtraction")

print("3. Multiplication")

print("4. Division")

print("5. Exit")

# Take input from the user

choice = input("Enter choice (1/2/3/4/5): ")

if choice == '5':

print("Exiting the calculator. Goodbye!")

break

# Check if the choice is valid

if choice in ('1', '2', '3', '4'):

num1 = float(input("Enter first number: "))

num2 = float(input("Enter second number: "))

# Perform the selected operation

if choice == '1':

print(f"Result: {num1} + {num2} = {add(num1, num2)}")

elif choice == '2':

print(f"Result: {num1} - {num2} = {subtract(num1, num2)}")

elif choice == '3':

print(f"Result: {num1} \* {num2} = {multiply(num1, num2)}")

elif choice == '4':

print(f"Result: {num1} / {num2} = {divide(num1, num2)}")

else:

print("Invalid choice! Please select a valid option.")

pg3

def is\_prime(num):

if num <= 1:

return False

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

if num % i == 0:

return False

return True

# Function to find the nth prime number

def nth\_prime(n):

count = 0 # To count the prime numbers found

num = 2 # Starting number to check for primes

while True:

if is\_prime(num):

count += 1

if count == n:

return num

num += 1

# Main part of the program

n = int(input("Enter the position of the prime number you want to find (n): "))

if n <= 0:

print("Please enter a positive integer greater than 0.")

else:

print(f"The {n}-th prime number is: {nth\_prime(n)}")

pg4

area\_square = lambda side: side \* side

# Lambda function to calculate the area of a rectangle (length \* width)

area\_rectangle = lambda length, width: length \* width

# Lambda function to calculate the area of a triangle (0.5 \* base \* height)

area\_triangle = lambda base, height: 0.5 \* base \* height

# Main loop for the menu-driven program

while True:

print("\nChoose a shape to calculate the area:")

print("1. Square")

print("2. Rectangle")

print("3. Triangle")

print("4. Exit")

# Get user choice

choice = input("Enter choice (1/2/3/4): ")

# Perform calculation based on choice

if choice == '1':

side = float(input("Enter the side length of the square: "))

print(f"The area of the square is: {area\_square(side)}")

elif choice == '2':

length = float(input("Enter the length of the rectangle: "))

width = float(input("Enter the width of the rectangle: "))

print(f"The area of the rectangle is: {area\_rectangle(length, width)}")

elif choice == '3':

base = float(input("Enter the base length of the triangle: "))

height = float(input("Enter the height of the triangle: "))

print(f"The area of the triangle is: {area\_triangle(base, height)}")

elif choice == '4':

print("Exiting the program. Goodbye!")

break

else:

print("Invalid choice! Please select a valid option.")

pg5

num\_terms = int(input("Enter the number of terms for powers of 2: "))

# Use map and lambda to calculate powers of 2

powers\_of\_2 = list(map(lambda x: 2 \*\* x, range(num\_terms)))

# Display the result

print(f"Powers of 2 up to {num\_terms} terms: {powers\_of\_2}")

pg6

def factorial(num):

if num == 0 or num == 1:

return 1

else:

result = 1

for i in range(2, num + 1):

result \*= i

return result

# Function to calculate the sum of the series

def sum\_series(n):

series\_sum = 0

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

term = (i \*\* i) / factorial(i)

series\_sum += term

return series\_sum

# Input: Number of terms in the series

n = int(input("Enter the number of terms: "))

# Display the sum of the series

print(f"The sum of the series up to {n} terms is: {sum\_series(n)}")

pg7

def compare(S1, S2, n):

# Check if the first n characters of both strings are the same

if S1[:n] == S2[:n]:

return True

else:

return False

# Main part of the program

# Take user inputs

S1 = input("Enter the first string (S1): ")

S2 = input("Enter the second string (S2): ")

n = int(input("Enter the number of characters to compare: "))

# Call the compare function and display the result

result = compare(S1, S2, n)

# Output the result

if result:

print("The first", n, "characters of both strings are the same.")

else:

print("The first", n, "characters of both strings are not the same.")

pg8

def factorial(num):

if num == 0 or num == 1:

return 1

else:

result = 1

for i in range(2, num + 1):

result \*= i

return result

# Function to calculate permutations p(n, r) = n! / (n - r)!

def permutations(n, r):

return factorial(n) // factorial(n - r)

# Function to calculate combinations c(n, r) = n! / (r! \* (n - r)!)

def combinations(n, r):

return factorial(n) // (factorial(r) \* factorial(n - r))

# Main part of the program

# Take user inputs

n = int(input("Enter the value of n (total objects): "))

r = int(input("Enter the value of r (objects taken at a time): "))

# Calculate and display the results

if r > n:

print("Invalid input: r should be less than or equal to n.")

else:

print(f"The number of permutations (P({n}, {r})) is: {permutations(n, r)}")

print(f"The number of combinations (C({n}, {r})) is: {combinations(n, r)}")