**Дороблені та виправленні завдання**

Кобець Анастасія 331 ФКНФМ

**Практичне заняття 3-4 ГРОШІ**

***Задача 1.*** *Молодші цифри степені*

def quick\_pow(a, n, d):

# Базовий випадок: a^0 = 1 (mod d)

if n == 0:

return 1 % d

else:

# Рекурсивно обчислюємо a^(n/2) mod d та зберігаємо результат в A

A = (quick\_pow(a, n // 2, d) \*\* 2) % d

# Якщо n парне, повертаємо A

if n % 2 == 0:

return A

# Якщо n непарне, повертаємо (A \* a) mod d

else:

return (A \* a) % d

# Введення даних з клавіатури

a = float(input("Введіть значення a: "))

n = int(input("Введіть значення n: "))

d = 100

# Обчислення та виведення результату

result = quick\_pow(a, n, d)

print(f"Результат обчислення ({a}^{n}) mod {d} = {result}")

*Результатом виконання буде:*

Введіть значення a: 5

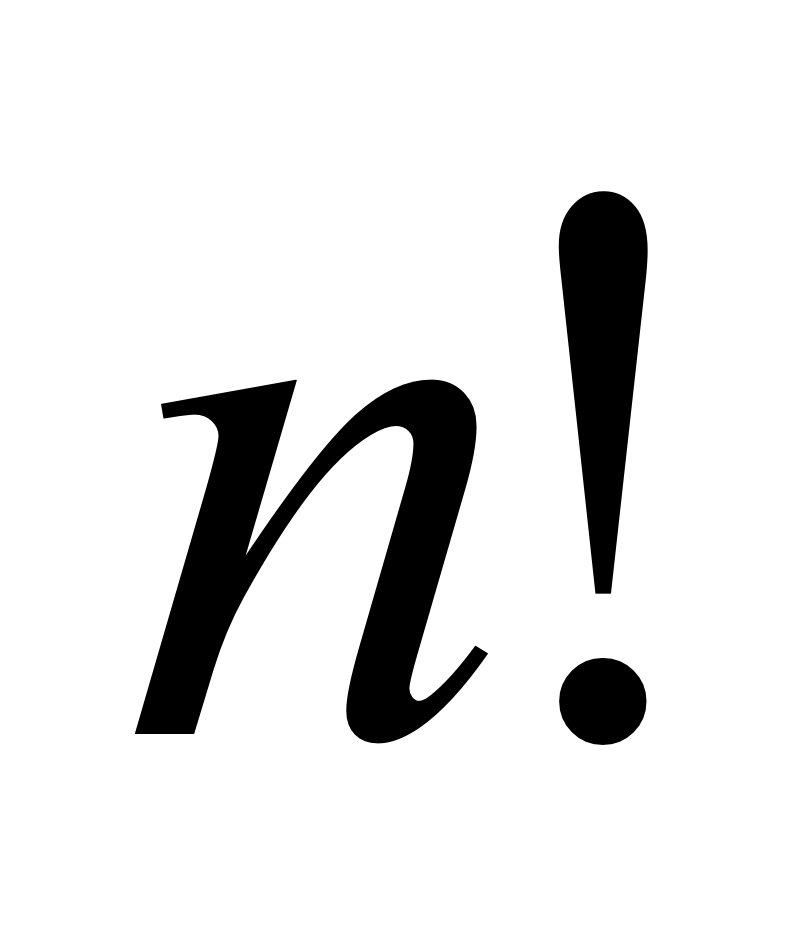
Введіть значення n: 5

Результат обчислення (5.0^5) mod 100 = 25.0

*\*в першому варіанті цієї задачі, що я зробила і надіслала було реалізовано неправильний алгоритм через підрахунок результату і просто поділене на 100*

**Практичне заняття 5-6 ФАКТОРІАЛ**

**Задача 3 1. Факторіал**

***Задача 1.*** *Знайти число нулів, якими закінчується число у десятковій системі числення.*

def count\_trailing\_zeros\_factorial(n):

count\_zeros = 0

# Перебираємо кожне число від 5 до n з кроком 5

for i in range(5, n + 1, 5):

current = i

# Поки число є кратним 5, додаємо до лічильника нульовий розряд

while current % 5 == 0:

count\_zeros += 1

current //= 5 # Зменшуємо число, щоб обробити всі його кратники 5

return count\_zeros

# Введення значення n з клавіатури

n = int(input("Введіть значення n: "))

# Обчислення та виведення кількості нулів на кінці n!

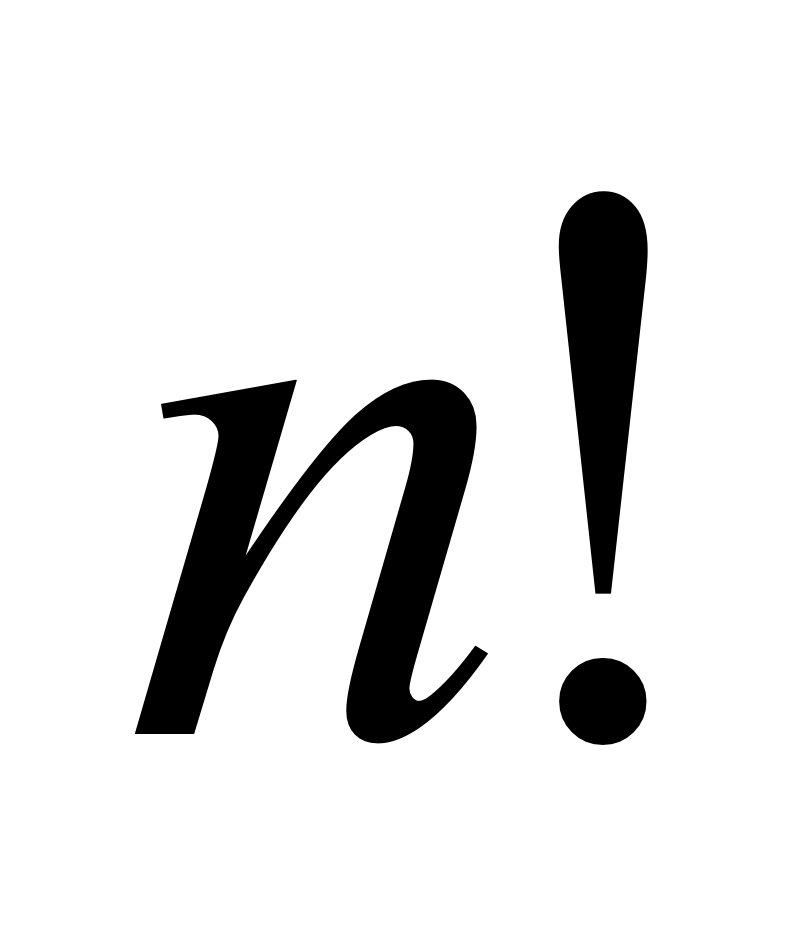
result = count\_trailing\_zeros\_factorial(n)

print(f"Кількість нулів на кінці {n}! = {result}.")

*Результатом виконання буде:*

Введіть значення n: 123

Кількість нулів на кінці 123! = 28

***Задача 2****. Знайти молодшу не рівну нулю цифру числа  у десятковій системі числення.*

def last\_nonzero\_digit\_factorial(n):

result = 1

# Обчислюємо факторіал числа n

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

result \*= i

# Знаходимо останню ненульову цифру

while result % 10 == 0:

result //= 10

return result % 10

# Приклад виклику функції

n = int(input("Введіть число n: "))

result = last\_nonzero\_digit\_factorial(n)

print(f"Остання ненульова цифра факторіала {n}!: {result}")

*Результатом виконання буде:*

Введіть число n: 123

Остання ненульова цифра факторіала 123!: 6

*\*в попередніх задачах було допущено помилку в виведені результату самого факторіала на екран та невірність алгоритму з заданою задачею*

***Задача Сума чисел.*** *Дано набір натуральних чисел x1, x2, … xn та натуральне число А. Знайти серед них такі два числа xi xj такі, що xi + xj = A чи показати що такої пари не існує.*

def find\_all\_pairs\_with\_sum(numbers, target\_sum):

pairs = []

numbers.sort() # Сортування масиву для використання зустрічного пошуку

left, right = 0, len(numbers) - 1

while left < right:

current\_sum = numbers[left] + numbers[right]

if current\_sum == target\_sum:

current\_pair = (numbers[left], numbers[right])

if current\_pair not in pairs:

pairs.append(current\_pair)

left += 1

right -= 1

elif current\_sum < target\_sum:

left += 1

else:

right -= 1

return pairs

# Введення набору чисел та цільової суми

numbers\_str = input("Введіть числа через пробіл: ")

numbers = list(map(int, numbers\_str.split()))

target\_sum = int(input("Введіть цільову суму: "))

# Знаходження всіх пар чисел

result\_pairs = find\_all\_pairs\_with\_sum(numbers, target\_sum)

# Виведення результату

if result\_pairs:

print(f"Пари чисел, які утворюють суму {target\_sum}:")

for pair in result\_pairs:

print(f"{pair[0]} та {pair[1]}")

else:

print(f"Пари чисел, які утворюють суму {target\_sum}, не існує.")

*Результатом виконання буде:*

Введіть числа через пробіл: 1 2 3 5 7 4 8 9 0

Введіть цільову суму: 6

Пари чисел, які утворюють суму 6:

1 та 5

2 та 4

*\*в цій задачі не було реалізовано сортування зустрічним пошуком, також додано сортування по зростанню і виводить на екран лише унікальні пари чисел*

**Практичне заняття 7-8 КАНТОРОВА МНОЖИНА**

***Задача 7. Канторова множина***

# Задані позитивні цілі числа a, b, n

a = int(input("Введіть значення a: "))

b = int(input("Введіть значення b: "))

n = int(input("Введіть значення n: "))

# Крок 1: Виділити цілу частину C числа a/b

C = a // b

# Крок 2: Виділити дробову частину D числа a/b

D = (a - C \* b) / b

# Крок 3: Перевести цілу частину в трійкову систему

p = 3

k = 1

c = [] # Список для зберігання цифр трійкового представлення цілої частини

while D > 0:

c.append(D % p)

D //= p

k += 1

# Крок 4: Перевести правильну дробову частину Q в трійкову систему

p = 3

q = [] # Список для зберігання цифр трійкового представлення дробової частини

for i in range(n):

d = a \* p

q.append(d // b)

a = d % b

# Перевірка на приналежність до множини Sn

if all(x < n for x in c) and all(x < n for x in q):

# Виведення позитивного результату

print(f"Належить множині Sn.")

else:

# Виведення негативного результату

print(f"Не належить множині Sn.")

*Результатом виконання буде:*

Введіть значення a: 7

Введіть значення b: 8

Введіть значення n: 14

Належить множині Sn.

*\*в цій задачі було додано хибне виведення результатів на екран з додаванням всіх підрахунків*

***Задача 8.3. Квадратне рівняння***

import math

# Введення коефіцієнтів a, b, c

a = int(input("Введіть коефіцієнт a: "))

b = int(input("Введіть коефіцієнт b: "))

c = int(input("Введіть коефіцієнт c: "))

# Обчислення дискримінанту

if (a,b,c) != 0 and a > 0:

D = b\*\*2 - 4 \* a \* c

#Перевірка, чи рівняння має розв'язки

if D < 0:

print(f"D<0")

if D == 0:

if b%(2\*a) == 0:

print(f"Розв'язки рівняння: x1,2 = {-b/(2\*a)}")

else:

print(f"Розв'язки рівняння: x1,2 = {-b}/{2\*a}")

if D > 0:

#знаходження кореня з дискримінанту та виділення цілої частини

n=D

e=0

while n > 0 and (math.sqrt(n) % 1 != 0 or D % n != 0):

n = n - 1

if n == 1:

n=math.sqrt(D)

e=int(D/n)

n=int(math.sqrt(D))

if e != 1:

if b == 0:

if n == 1:

if 2\*a == 1:

print(f"±√{D}")

else:

print(f"(±√{D}) / {2\*a}")

else:

if 2\*a == 1:

print(f"(±√{D}) / {2\*a}")

else:

print(f"(±{n}√{e}) / {2\*a}")

elif n == 1:

if 2\*a == 1:

print(f"{-b}±√{D}")

else:

print(f"({-b}±√{D}) / {2\*a}")

else:

if 2\*a == 1:

print(f"{-b}±√{D}")

else:

print(f"({-b}±√{D}) / {2\*a}")

if e == 1:

if b == 0:

if n == 1:

if 2\*a == 1:

print(f"±{n}")

else:

print(f"(±{n}) / {2\*a}")

else:

if 2\*a == 1:

print(f"(±{n}) / {2\*a}")

else:

print(f"(±{n}) / {2\*a}")

elif n == 1:

if 2\*a == 1:

print(f"{-b}±{n}")

else:

print(f"({-b}±{n}) / {2\*a}")

else:

if 2\*a == 1:

print(f"{-b}±{n}")

else:

print(f"({-b}±{n}) / {2\*a}")

else:

print("Рівняння не має розв'язків")

else:

print(f"Не виконані умови a>0 або a,b,c!=0»

*Результатом виконання буде:*

Введіть коефіцієнт a: 1

Введіть коефіцієнт b: 6

Введіть коефіцієнт c: -3

(-6±4√3) / 2

*\* в попередньому варіанті мій файл з завданнями не прогрузився повністю і повної умови з урахуванням всіх випадків не було реалізовано. змінено задачу відповідно до умови і враховуються всі випадки квадратного рівняння. також було особисто розроблено функцію з вирахуванням цілої частини з під кореню*

***Задача Алгебра числових сегментів.***

def merge\_segments(segments):

# Сортуємо числові сегменти за початковими точками

sorted\_segments = sorted(segments, key=lambda x: x[0])

merged\_segments = []

current\_segment = sorted\_segments[0]

for segment in sorted\_segments[1:]:

# Перевіряємо, чи поточний сегмент перетинається з наступним

if current\_segment[1] >= segment[0]:

# Якщо так, об'єднуємо їх

current\_segment = [current\_segment[0], max(current\_segment[1], segment[1])]

else:

# Якщо не перетинається, додаємо поточний сегмент до об'єднаних та оновлюємо поточний сегмент

merged\_segments.append(tuple(current\_segment))

current\_segment = segment

# Додаємо останній об'єднаний сегмент

merged\_segments.append(tuple(current\_segment))

return merged\_segments

# Запитуємо користувача про введення числових сегментів

num\_segments = int(input("Введіть кількість числових сегментів: "))

segments = []

for i in range(num\_segments):

start = float(input(f"Введіть початок {i + 1}-го сегмента: "))

end = float(input(f"Введіть кінець {i + 1}-го сегмента: "))

segments.append((start, end))

# Викликаємо функцію та виводимо результат

result = merge\_segments(segments)

print("Результат об'єднання числових сегментів:", result

*Результатом виконання буде:*

Введіть кількість числових сегментів: 5

Введіть початок 1-го сегмента: 1

Введіть кінець 1-го сегмента: 3

Введіть початок 2-го сегмента: 2

Введіть кінець 2-го сегмента: 5

Введіть початок 3-го сегмента: 6

Введіть кінець 3-го сегмента: 8

Введіть початок 4-го сегмента: 7

Введіть кінець 4-го сегмента: 10

Введіть початок 5-го сегмента: 12

Введіть кінець 5-го сегмента: 15

Результат об'єднання числових сегментів: [(1.0, 5.0), (6.0, 10.0), (12.0, 15.0)

*\*цієї задачі не було виконано попередньо, через помилку в моєму файлі з завданням*