КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра програмних систем і технологій

Т.В. Ковалюк

Методичні вказівки до лабораторних занять з дисципліни «Функціональне програмування»

для студентів спеціальності 121 «Інженерія програмного забезпечення»
освітнього рівня «бакалавр»

Київ 2020

Зміст

Загальні теоретичні відомості

Характеристика функціональних мов програмування

Основним поняттям функціонального програмування є функція. Функціональна програма є набором визначень функцій. Функції визначаються через інші функції або рекурсивно через самих себе. У процесі виконання програми функції отримують параметри, обчислюють і повертають результат, у випадку необхідності обчислюються значення інших функцій. Програмуючи функціональною мовою, програміст не повинен описувати порядок обчислень. Йому необхідно просто описати бажаний результат у вигляді системи функцій.

Строго функціональне програмування не має операцій присвоєнь та засобів передачі керування. Повторні обчислення здійснюються за допомогою рекурсії, яка є основним засобом функціонального програмування.

Серед важливих властивостей функціонального програмування є такі:

1. Представлення програми і даних відбувається через списки. Це дає змогу програмі обробляти інші програми і навіть саму себе.
2. Функціональні мови, як правило, є інтерпретуючими мовами.
3. Функціональні мови є безтиповими, це означає, що символи не зв'язуються за замовчуванням з яким-небудь типом.
4. Функціональні мови мають незвичний синтаксис через велику кількість дужок.
5. Функціональні програми, написані для обробки символьних даних, є набагато коротші, аніж написані імперативними мовами/

Функції-примітиви мови Scheme

Вважається, що строго функціональна мова програмування має дуже обмежене число базових функцій, на основі яких можна побудувати всі інші функції. Як правило, вибирається сім примітивних функцій. Ця обмеженість є важливою в теоретичному плані в галузі програмування, оскільки вирішується проблема розв'язання довільної задачі з допомогою обмеженого набору примітивних функцій.

Виклик функції у функціональних мовах програмування здійснюється у формі списку і має такий формат:

(function_name arg1 arg2 ... argN),

Де **function_name – ім'я функції,**

arg1,arg2,...argN – її аргументи.

Базові функції функціональної мови програмування Scheme наведено в табл. 1.

Таблиця 1. Базові функції функціональних мов програмування

Мова Scheme	**Семантика**
QUOTE	Функція використовується для блокування обчислень, що призводить до інтерпретації частини S-виразу не як програми, а як даних
(CAR object)	Вибирає голову непорожнього списку, пари або неправильно-сформованого списку
(CDR object)	Вибирає хвіст непорожнього списку, пари або неправильно-сформованого списку
(CONS object1 object2)	Функція-конструктор - об'єднує об'єкти 1 та 2
(EQ? atom1 atom2)	Порівнює два об'єкта на тотожність
LIST? object)	Перевіряє чи є object списком.
(PAIR? object)	Перевіряє чи є object парою

(COND (condition1 action1) (condition2 action2)… (conditionN actionN))	Набуває значення з множини значень action1, action2, ..., actionN в залежності від значень умовних виразів condition1, condition2, …, conditionN.

Таблиця 2. Деякі математичні функції

Мова Scheme	Семантика
ABS	функція знаходить модуль числа.
EXP	функція, яка повертає e^x.
LN	функція, яка повертає натуральний логарифм аргументу.
SIN	функція, яка повертає синус аргументу.
COS	функція, яка повертає косинус аргументу.
SQRT	функція, яка повертає квадратний корінь аргументу.

Таблиця 3. Предикати

NUMBER?	предикат, який перевіряє чи аргумент є числом
POSITIVE?	предикат, який перевіряє чи аргумент є додатнім числом.
NEGATIVE?	предикат, який перевіряє чи аргумент є від'ємним числом
EVEN?	предикат, який перевіряє чи число є парним.
ODD?	предикат, який перевіряє, чи число є непарним.
NULL?	предикат, який перевіряє чи аргумент є порожнім списком.

Середовище функціонального програмування Dr.Racket

Оскільки в середовищі Dr.Racket підтримується ряд діалектів та стандартів, виберемо класичний стандарт Scheme R5RS. Для цього потрібно обрати меню Мова-> Обрати мову ->Other languages-> R5RS.

На рис. 1 наведено вигляд головного вікна середовища.

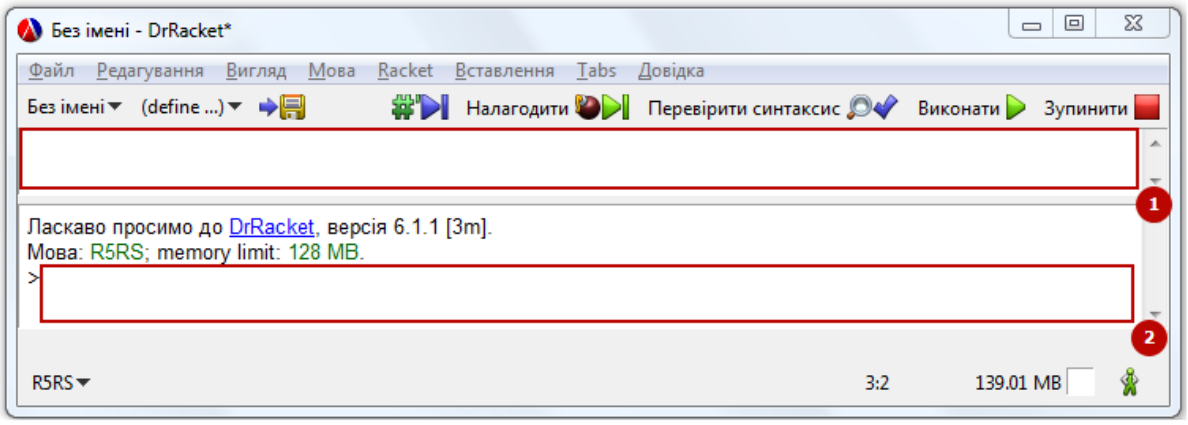


Рис. 1. Головне вікно середовища.

Головне вікно середовища розробки поділене на дві частини: 1 – область для вводу тексту програми; 2 – область командного рядка. В першій частині зручно писати текст програми і всі виклики, в другій – виклики функцій.

На панелі інструментів винесено кнопки для від лагодження, перевірки синтаксису, запуску на виконання та зупинки програми (рис. 2).

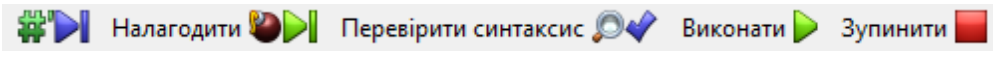


Рис. 2. Панель інструментів в середовищі Dr.Racket

Приклад 1. Обчислимо суму 1 + 2. На мові Scheme потрібно написати (+ 1 2). Після цього запустимо програму на виконання клавішою F5 або кнопкою «Запустити». Результат виконання програми відобразиться в другій половині екрану (рис. 3).

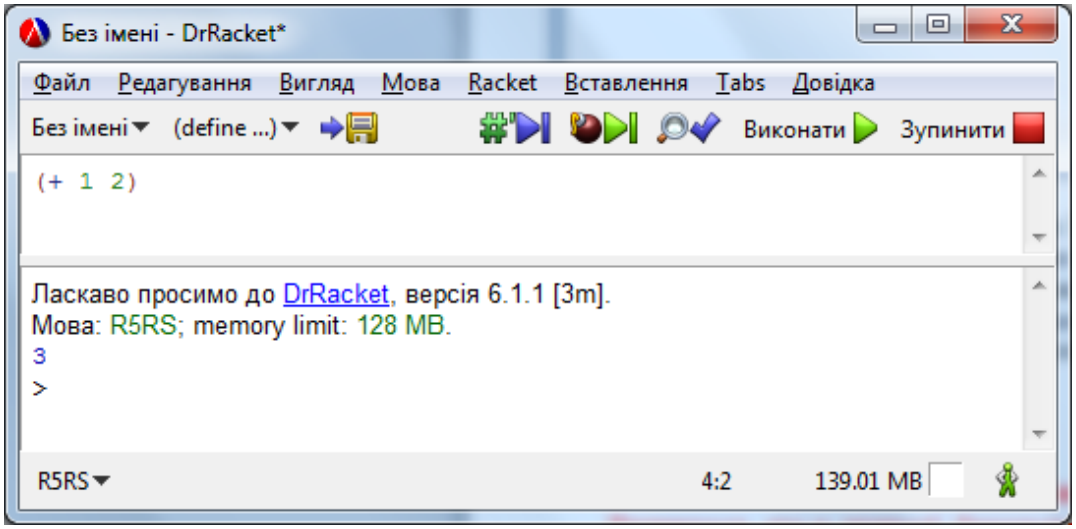


Рис. 3. Виконання програми в середовищі Dr. Racket

Є можливість виконувати команди безпосередньо в командному рядку: після введеного тексту програми потрібно натиснути клавішу Enter. Перемножимо числа 2, 3, 4: для цього введемо в командний рядок текст (* 2 3 4) і натиснемо клавішу Enter. Наступний рядок міститиме шуканий результат (рис. 4).

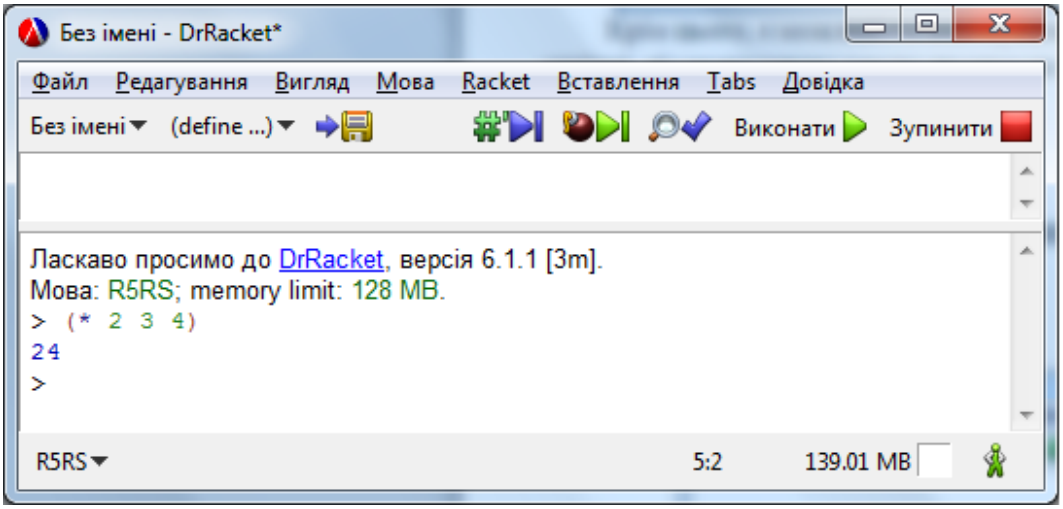


Рис. 4. Виконання програми в командному рядку

Щоб поставити коментар потрібно перед текстом поставити символ «;». Scheme не є чутливим до регістру і до кількості пробілів між об'єктами. Вирази (a) та (a) є тотожними.

На рис. 5 наведено приклад використання змінних та коментарів.

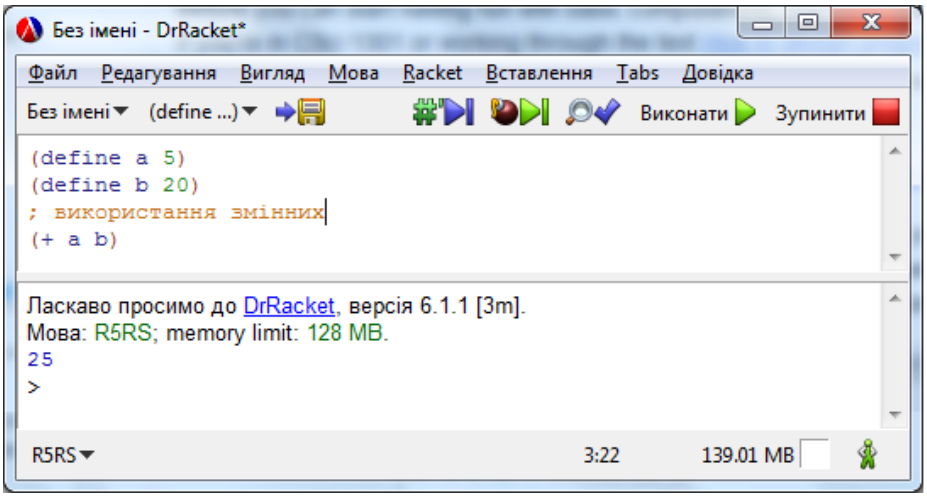


Рис. 5. Використання змінних та коментарів у середовищі Dr. Racket

Процес налаштування коду в середовищі Dr.Racket

Приклад 2. Обчислимо вираз: **(+ 2 (* 3 4))**. Замість кнопки «Виконати» натиснемо кнопку «Налагодити». Після цього головне вікно буде мати вигляд як на рис. 6.

Вікно має такі основні частини: 1 – текст програми, 2 – поточний результат виконання програми, 3 – меню для налаштування, 4 – вікно із відображенням вмісту стеку, 5 – вікно для відображення поточних значень змінних.

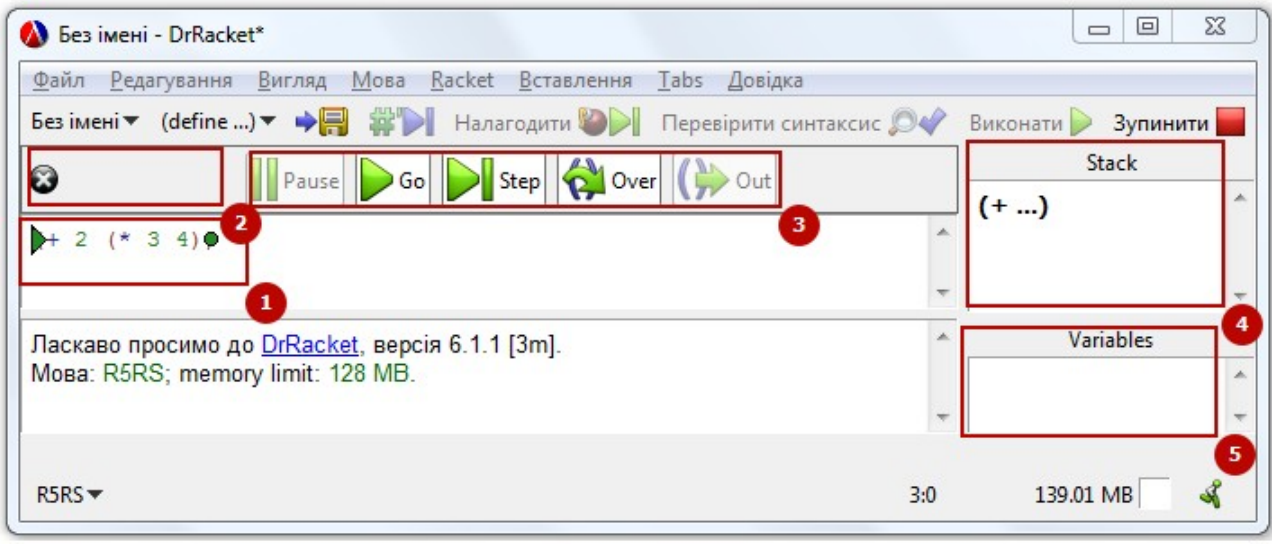


Рис. 6. Інтерфейс середовища Dr.Racket у режимі відлагодження

Спочатку в стеку є наявна функція додавання (рис. 14). Курсор відлагодження вказує на цю функцію. Натиснемо кнопку «Step». В стеку з'явилась функція множення (рис. 7). Курсор налаштування вказує на цю функцію у вікні тексту програми.

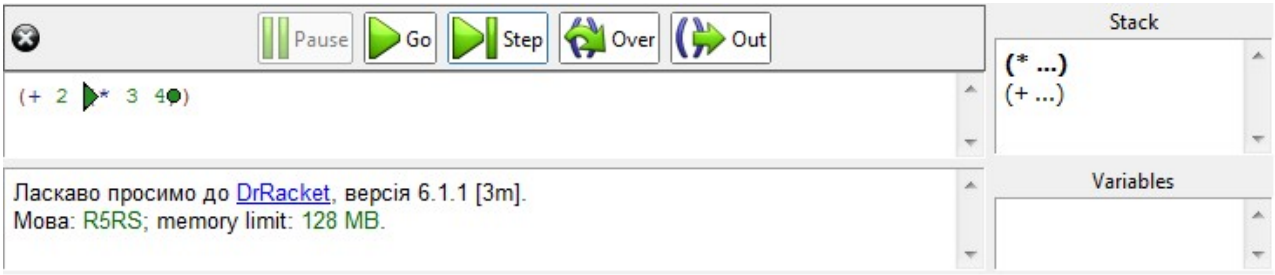


Рис. 7. Додавання в стек функції множення

Натиснемо ще раз кнопку «Step». Оскільки далі за функцією множення немає інших функцій, відбудеться обчислення множення за вказаними параметрами 3 і 4. Проміжний результат (число 12) відображено у відповідній частині екрану (рис. 16).

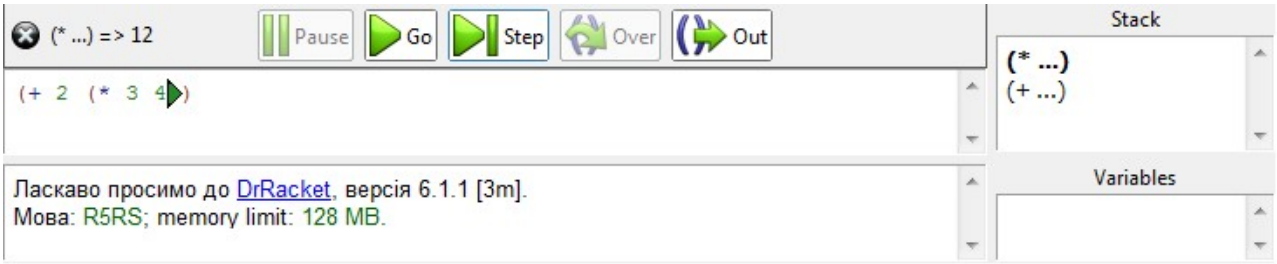


Рис. 8. Здійснення обрахунків для функції множення

Натиснемо ще раз кнопку «Step». Як видно із стеку, функція множення виконалась і вийшла з стеку. Курсор налаштування вказує на кінець виклику функції додавання. Результат виконання функцій (число 14) відображено в області біля меню налаштування (рис. 9).

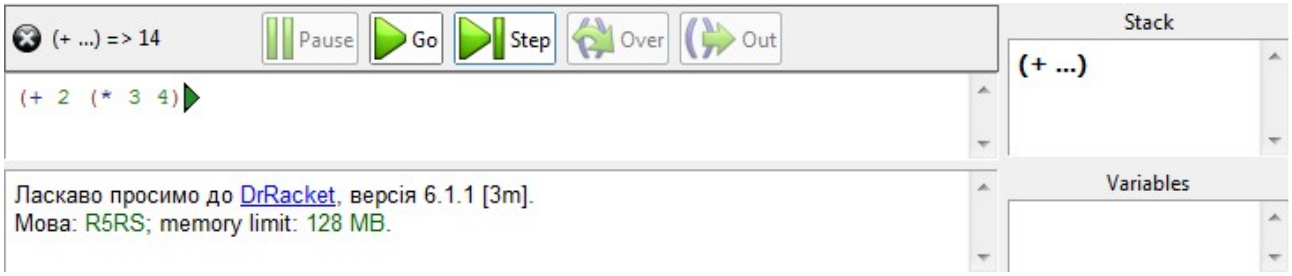


Рис. 9. Здійснення обрахунків для функції додавання

Чергове натискання кнопки «Step» завершує виконання програми – з стеку вивільняється функція додавання, а остаточний результат відображається у вікні командного рядка (рис. 18).

Рис. 18. Завершення виконання налаштування програми

Лабораторна робота 1.
Використання рекурсії для організації повторювальних процесів

Мета

Сформувати декларативне мислення в галузі програмування завдяки використанню чистих функцій, рекурсій замість циклів, запобіганню даних, що змінюються. Опанувати застосування рекурсивних функцій для обчислювальних процесів.

Рейтингова таблиця лабораторної роботи 1

Вид роботи	**Кількість балів**	Deadline
Код задачі 1	2	жовтень
Код задачі 1	2	
Звіт	1	
Якість		
Разом	**5**	

Теоретичні відомості

Визначення процедур

Для створення функції з іменем потрібно використати один із форматів опису функції:

(define <name> (lambda (<arguments>) (<body>))),

або

(define (<name> <arguments>) (<body>)),

де **<name>** - ім'я функції; **<arguments>** - список аргументів (через пробіл); **<body>** - правильний S-вираз, який набуває значення.

Приклад.

```
(define (f x)
    (+ x 42))
Виклик функції: (f 23)
Результат  65
```

Процедура, в попередньому абзаці, є абстракцією виразу за допомогою об'єктів. У першому визначенні прикладу визначена процедура, названа f. (Зверніть увагу на круглі дужки навколо f x, що позначають, що це -визначення процедури.) Вираз (f 23) є викликом процедури, приблизно означає "вирахувати (+ x 42) (тіло процедури) з x, прив'язаним до 23 ".

Оскільки процедури є об'єктами, їх можна передавати в інші процедури:

```
(define (f x)
   (+ x 42))

(define (g p x)
        (p x))

Виклик : (g f 23)
Результат: 65
```

У цьому прикладі тіло g обчислюється з p, прив'язаним до f, і x, що прив'язаний до 23, що еквівалентно (f 23) і обчислюється в 65.

Фактично багато операцій Scheme забезпечуються не синтаксисом, а змінними, значеннями яких є процедури. Операція +, наприклад, в Scheme є всього лише регулярним ідентифікатором, пов'язаним з процедурою, що додає числові об'єкти. Те саме стосується і *, і багатьох інших:

```
(define (h op x y)
     (op x y))
```

(h + 23 42) =) 65
(h * 23 42) =) 966

Хвостова рекурсія

Це випадок рекурсії, коли рекурсивний виклик функції відбувається наприкінці її роботи. Використовується у мовах програмування для оптимізації, через можливість заміни виклику функції на ітерацію, без використання стеку. Ця оптимізація широко використовується у функціональних мовах програмування.

Коли відбувається виклик функції комп'ютер має запам'ятати адресу повернення, щоб після завершення викликаної функції повернутися і продовжити виконання програми. Зазвичай адреса виконання зберігається у стеку. Іноді, остання дія функції після завершення всіх інших операцій, це просто виклик функції, можливо самої себе, і повернення результату. В цьому випадку немає необхідності запам'ятовувати адресу повернення, нововикликана функція буде повертати результат безпосередньо за адресою повернення записаною для початкової функції.

Приклад на Scheme:

```
(define (factorial n)
  (define (fac-times n acc)
    (if (= n 0)
        acc
        (fac-times (- n 1) (* acc n))))
  (if (< n 0)
      (display "Невірний параметр!")
      (fac-times n 1)))
\
```

Звичайний рекурсивний спосіб обчислення факторіала, наведений нижче, **не** є хвостовим-рекурсивним, оскільки в кожному виклику функції після рекурсивного виклику виробляються додаткові операції, а саме множення на *n*.

Приклад на Scheme:

```
( define ( factorial n )
    ( if ( = n 0 )
       1
     ( * n ( factorial ( - n 1 ) )
) ) )
```

Зміст звіту

1. Титульний лист
2. Мета роботи
3. Умова завдання
4. Аналіз задачі та математичні забезпечення для розв’язання (у випадку математичної задачі)
5. Обґрунтування вибору мови функціонального програмування та IDE
6. Код
7. Скріншот роботи та результату
8. Оцінка достовірності результату (перевірка правильності результату)
9. Висновок

Завдання до лабораторної роботи 1

Написати процедури, що обчислюють задану функцію за допомогою рекурсивного процесу. Продемонструвати застосування звичайної та хвостової рекурсії.

Номер варіанта	Умова завдань
1.	1.1 Ввести з клавіатури натуральне число n. Знайти суму його цифр, використовуючи рекурентне означення функції $f(n)$: $f(n) = \begin{cases} n \bmod 10 + f(n/10), & якщо \quad n \neq 0 \\ 0, & якщо \quad n \neq 0 \end{cases}$ *Умова продовження рекурсії:* сума цифр числа дорівнює останній цифрі плюс сума цифр числа без останньої цифри (числа, що ділиться без остачі на 10). *Умова закінчення рекурсії:* якщо число дорівнює 0, то сума його цифр дорівнює 0.
	1.2. Вкладник поклав в банк sum грошових одиниць під pr відсотків за один період. Усі дані вводити з клавіатури. Використовуючи рекурсію, визначити величину вкладу по звершенні m періодів часу. Контрольний тест: введені дані: сума вкладу 1000, відсотки за період 1.25, кількість періодів 12, отриманий результат: 1160.75
2.	2.1 Ввести з клавіатури два натуральних числа n та m. Обчислити кількість комбінацій з n різних елементів по m. Кількість комбінацій визначається рекурентним співвідношенням: $C_n^k = \begin{cases} C_{n-1}^{k-1} + C_{n-1}^k, & n > 0 \\ 1, & k = n \geq 0 \ \ або \ \ k = 0, n < 0 \\ 0, & в\ них\ випадках \end{cases}$ де C_n^k біноміальні коефіцієнти, які розраховують за формулою $C_n^k = \frac{n!}{k!(n-k)!}$ Визначити глибину рекурсії.
	2.2 Увести з клавіатури два цілих числа A і B. Використовуючи рекурсію, вивести всі числа від A до B включно в порядку спадання, якщо A більше B або в порядку зростання в іншому випадку. Контрольний тест: введені числа 7 3, отриманий результат: 7 6 5 4 3.
3.	3.1. Ввести з клавіатури два натуральних числа n та m. Розрахувати значення функції Аккермана $A(m, n)$ та глибини рекурсії, використовуючи рекурентне співвідношення: $A(m/n) = \begin{cases} n+1, & якщо \quad m = 0 \\ A(m-1,\ 1), & якщо\ m > 0, n = 0 \\ A(m-1, A(m, n-1)), & якщо\ \mathrm{m} > 0, \mathrm{n} > 0 \end{cases}$ Визначити глибину рекурсії. Контрольний тест: $A(2, 2) = 7$.
	3.2 Увести з клавіатури натуральне число n >1. Вивести всі прості дільники цього числа в порядку неспадання з урахуванням кратності. Алгоритм повинен мати складність O ($\sqrt{n}$). Контрольний тест: введено число 18, отриманий результат: 2 3 3.
4.	4.1 Ввести з клавіатури два цілих додатних числа X та N. Звести число X в степінь N, використавши рекурентне співвідношення: $X^n = \begin{cases} X, & якщо \quad n = 1 \\ (X^{n/2})^2, & якщо\ n - парне \\ (X^{(n-1)/2})^2, & якщо\ n - непарне \end{cases}$ Визначити глибину рекурсії.
	4.2 Увести з клавіатури натуральне число n. Обчисліть суму його цифр. Контрольний тест: введено 179, отриманий результат: 17.

5.	5.1 Ввести з клавіатури два натуральних числа m та n. Знайти їх найбільший спільний дільник, застосувавши алгоритм Евкліда (GCD - Greatest Common Divisor) для рекурентного співвідношення: $GCD = \begin{cases} a, & b = Ka \\ b, & a = kb \\ GCD(a, b \bmod a,), & b \ge a \\ GCD(a \bmod b,\ b), & a \ge b \end{cases}$ Запис $a \bmod b$ означає остачу від ділення a на b. K - натуральне число, що означує кратність чисел a та b. Визначивши глибину рекурсії.
	5.2 Увести з клавіатури натуральне число n. Вивести всі його цифри по одній в зворотному порядку, розділяючи їх пробілами або новими рядками. При розв'язанні цього завдання дозволена тільки рекурсія і цілочислова арифметика. Контрольний тест: введено число 123, отриманий результат: 3 2 1.
6.	6.1. З n солдатів, вишикуваних в шеренгу, потрібно відібрати кількох в розвідку. Для здійснення цього виконується наступна операція: якщо солдат в шерензі більше ніж 3, то видаляються всі солдати, які стоять на парних позиціях, або всі солдати, які стоять на непарних позиціях. Ця процедура повторюється до тих пір, поки в шерензі залишиться 3 або менше солдатів. Їх і відсилають в розвідку. Обчислити кількість способів, якими можуть бути сформовані групи розвідників рівно з трьох осіб. Кількість солдатів n вводиться з клавіатури. Рекурентне співвідношення для обчислення кількості способів $f(n)$, якими можна сформувати групи розвідників з n осіб в шерензі, таке: $f(n) = \begin{cases} 0, \ якщо\ n = 1, n = 2 \\ 1, \ якщо\ n = 3 \\ 2 \times f(n/2), якщо\ n - парне \\ f(n/2) + f(n/2+1), якщо\ n - непарне \end{cases}$ Визначити глибину рекурсії.
	6.2 Увести з клавіатури натуральне число n. Вивести всі його цифри по одній в прямому порядку, розділяючи їх пробілами або новими рядками. При розв'язанні цього дозволена тільки рекурсія і цілочислова арифметика. Контрольний тест: введено число 123, отриманий результат: 1 2 3.
7.	7.1. Ввести з клавіатури два цілих додатних числа m>0 та n>0. Розробити рекурсивну функцію, яка обчислює площу трикутника на основі рекурентного співвідношення: $S(n,m) = \begin{cases} 1, & якщо\ n = m = 1 \\ S(n-1,m) + m, & якщо\ n > 1 \\ S(n,m-1) + 1, & якщо\ m > 1 \end{cases}$ Визначити глибину рекурсії.
	7.2 Увести з клавіатури натуральне число n > 1. Перевірити, чи є воно простим. Програма повинна вивести слово YES, якщо число просте і NO, якщо число складене. Алгоритм повинен мати складність O ($\sqrt{n}$). Для застосування рекурсії для цієї задачі потрібно робити рекурсію за параметром, яким є дільник числа. Контрольний тест: введено число 6, отриманий результат: NO.
8.	8.1 Ввести з клавіатури два цілих додатних числа a та b. Обчислити добуток двох цілих додатних чисел за рекурентним співвідношенням: $p = a \times b = \begin{cases} a, & якщо\ b = 1 \\ b, & якщо\ a = 1 \\ 0, \ якщо\ a = 0\ або\ b = 0 \\ a + p(a, b-1), в\ \text{інших випадках} \end{cases}$ Визначити глибину рекурсії.
	8.2.Увести з клавіатури два натуральних числа k та s. Визначити кількість k-значних натуральних чисел, сума цифр яких дорівнює s. Запис натурального числа не може починатися з цифри 0. В цьому завданні можна використовувати цикл для перебору всіх цифр, що стоять на будь-якій позиції. Контрольний тест: введені числа 3 15, отриманий результат: 69.

9.	9.1. По колу стоять n людей, яким присвоєні номери від 1 до n. Починаючи відлік з першого і рухаючись по колу, кожна друга людина виходитиме з кола доти, поки не залишиться одна. Нехай номер того, хто залишився, x. Потім по колу стоятиме x людей і процедура виходу з колу людей повторюватиметься доти, поки не залишиться одна людина з номером y. Ці процедури повторюватимуться доти, поки номер тої людини, що залишиться, не стане рівним первинній кількості людей в потоковому раунді. Визначити номер людини, яка залишилася, і кількість повторів процедури. Номер людини $f(n)$, що залишилася, обчислюється за рекурентним співвідношенням: $f(n)=\begin{cases}1, & якщо \quad n=1\\ 2\times f(n/2)-1. & якщо \quad n-парне\\ 2\times f(n/2)+1, & якщо \quad n-непарне\end{cases}$ Визначити глибину рекурсії.
	9.2. Увести з клавіатури натуральне число n. Вивести слово YES, якщо число n є точним степенем двійки, або слово NO в іншому випадку. Операцією зведення в степінь користуватися не можна!. Контрольний тест: уведено число 8, отриманий результат: YES.
10.	10.1. Біноміальні коефіцієнти розкладання бінома $(a+b)^i$ утворюють i-й рядок трикутника Паскаля. Кожне число в трикутнику, крім перших трьох, є сумою чисел, розташованих над ним у попередньому рядку. Число в i-му рядку (i = 0, 1, 2, …) на j-му місці (j = 0, 1, …, i) задається формулою $C_n^k=\frac{i!}{j!(i-j)!}$. Сам трикутник Паскаля задається рекурентним співвідношенням. $f(i,j)=\begin{cases}1, & i=0, j=0\\ 1, & i=1, j=0\\ 1, & j=1, j=1\\ f(i,j-1)+f(i-1,j), & в \ інших \ випадках.\end{cases}$ 1 1 1 1 2 1 1 3 3 1 1 4 6 4 1 «Верхівка» трикутника має наступний вигляд. Надрукувати перші n рядків трикутника Паскаля, використавши рекурсивне визначення його елементів. Кількість рядків вводиться з клавіатури. Визначити глибину рекурсії.
	10.2 Увести з клавіатури два натуральних a і b. Визначити кількість послідовностей з a нулів і b одиниць, в яких ніякі два нулі не стоять поруч. Вивести знайдені послідовності з нулів та одиниць. Контрольний тест: введені числа 2 2, отриманий результат: 3 0101 1010 0110.
11.	11.1.Послідовність 1, 1, 2, 3, 5, 8, ... складається з чисел Фібоначчі. Кожен елемент, починаючи з третього, дорівнює сумі двох попередніх. Рекурентне співвідношення для розрахунку чисел Фібоначчі таке: $f(n)=\begin{cases}0, & n=0\\ 1, & n=1\\ f(n-1)+f(n-2), & n\ge 2\end{cases}$ Ввести з клавіатури два натуральних числа m та n, які означають кількість чисел та номер числа в послідовності Фібоначчі. Вивести послідовність чисел Фібоначчі в кількості m елементів та значення n-го числа. Передбачити випадок $m < n$. Визначити глибину рекурсії.
	11.2 Створити рекурсивну функцію, яка отримує числа, зчитуючи їх з клавіатури, і перевіряє їх на непарність. Кінець вводу - число 0. Функція не повертає значення, а відразу ж виводить результат на екран, зберігаючи порядок ведених чисел. У цьому завданні не можна використовувати глобальні змінні і передавати будь-які аргументи в рекурсивну функцію. Основна програма повинна складатися тільки з виклику цієї функції. Контрольний тест: введені числа 3 2 1 0, отриманий результат: 3 1.
12.	Ввести з клавіатури два натуральних числа n та k. Кількість розміщень без повторень A_n^k і кількість сполучень без повторень C_n^k можуть бути знайдені відповідно за формулами $A_n^k=\frac{n!}{(n-k)!}$, $C_n^k=\frac{n!}{n!(n-k)!}$. Застосувати формули для розрахунку кількості розміщень і сполучень для визначення кількості способів розподілу k обов'язків між n членами комісії та кількості способів розподілу уроків в n класах між трьома вчителями, якщо кожен учитель викладатиме у k класах. Визначити глибину рекурсії.

	12.2. Увести з клавіатури натуральне число n, десятковий запис якого не містить нулів. Отримайте число, записане тими самими цифрами, але в протилежному порядку. При розв'язанні цього завдання не можна використовувати цикли, рядки, списки, масиви, дозволяється тільки рекурсія і цілочислова арифметика. Функція повинна повертати ціле число, яке є результатом роботи програми, виводити число по одній цифрі не можна. Контрольний тест: введено число 179, отримали результат: 971.
13.	13.1 Ввести з клавіатури два натуральних числа для обчислення кореня k>1 -ого степеню з числа a>=0, використовуючи рекурентну формулу $sqrt(a,k,n)=\begin{cases}1, & n=0, a\ge 0, k>1\\ \frac{1}{k}\left((k-1)x_n+\frac{a}{x_n^{k-1}}\right), & n>0, a\ge 0, k>1\end{cases}$, де n - кількість елементів послідовності наближених значень кореня k-ого степеню з числа a. Визначивши глибину рекурсії.
	13.2 Увести з клавіатури два натуральних числа m в десятковій системі числення та n в двійковій системі. Використовуючи рекурсивну функцію, перевести число m з десяткової системи числення в двійкову, а число n з двійкової системи числення в десяткову. Контрольний тест: введені числа 25 10101, отриманий результат 11001 21.
14.	14.1. Увести з клавіатури три натуральних числа b, p, m. Обчислити значення виразу $b^p \bmod m$, де операція mod обраховує остачу від ділення цілих чисел. Для зведення в степень b^p з логарифмічною складністю $O(\log p)$ використати рекурентне співвідношення: $b^p=\begin{cases}1, & p=0\\ (b^{p/2})^2, & p-парне\\ b\times(b^{p/2})^2, & p-непарне\end{cases}$ Визначити глибину рекурсії.
	14.2. Увести з клавіатури натуральне число n. Використовуючи рекурсивну функцію, визначити і вивести всі непарні (парні) числа з послідовності цілих чисел від n до 0, зберігаючи їх порядок. Контрольний тест: введено число 8, отриманий результат: 7 5 3 1.
15.	15.1. У речовій лотереї розігруються m предметів. Усього в урні n квитків. Виймається k квитків. Значення m, n, k вводять з клавіатури. Скількома способами квитки можна вийняти з урни так, щоб: а) рівно два з них були виграшними, б) принаймні два з них були виграшними? Кількість способів вибору квитків визначається формулою сполучень $C_n^k=\frac{n!}{k!(n-k)!}$, комбінаторними правилами суми та добутку. Реалізувати рекурсивний варіант розв'язку задач. Визначивши глибину рекурсії.
	15.2. Монотонною послідовністю називається послідовність натуральних чисел, в якій кожне натуральне число k зустрічається рівно k раз: 1, 2, 2, 3, 3, 3, 4, 4, 4, 4 ... Ввести з клавіатури натуральне число n. Використовуючи рекурсію, вивести перші n членів цієї послідовності. Контрольний тест: введено число 15, отриманий результат: 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5.
16.	16.1. Визначити кількість розбиття $p(n,m)$ додатного цілого числа n, значення якого ввести з клавіатури. Розбиття цілого числа – це його зображення у вигляді суми цілих додатних чисел. Кількість розбиття цілого числа n із доданками, що не перевищують значення m, визначається за рекурентним співвідношенням: $p(n,m)=\begin{cases}0, & якщо\ m=0, n<0\\ 1, & якщо\ m=1, n=0\\ 1+p(n.m-1), & якщо\ n=m\\ p(n,m-1+p(n-m,m), & otherwise\end{cases}$ Реалізувати рекурсивний варіант розв'язку задачі, Визначивши глибину рекурсії.
	16.2. Увести з клавіатури два натуральних m і n. Використовуючи рекурсію, визначити найменше спільне кратне (НСК) введених чисел. НСК двох цілих чисел m і n є найменше натуральне число, яке ділиться на m і n без залишку, тобто кратне їм обом. НСК(m, n)=(m* n)/НСД(m, n), де НСД - найменший спільний дільник. Контрольний тест: введені числа 16 20, отриманий результат: 80.

17.	17.1. Увести з клавіатури два цілих числа *p, q*. Обчислити суму функцій *f(n)* за введеними значеннями *p, q* за формулою $S(p,q)=\sum_{n=p}^{q} f(n)$. Функції *f(n)* при *n*, що змінюється від *p* до *q*, визначена через рекурентне співвідношення: $f(n)=\begin{cases} n\%10, & якщо\ n\%10>0 \\ 0, & якщо\ n=0 \\ f(n/10), & otherwise \end{cases}$ Реалізувати рекурсивний варіант розв'язку задачі. Визначити глибину рекурсії.
	17.2. Увести з клавіатури натуральне число *n*. Вивести всі прості множники цього числа в довільному порядку з урахуванням кратності. Контрольний тест: уведено число 84, отриманий результат: 2 2 3 7.
18.	18.1. Числа Фібоначчі визначаються рекурентним співвідношенням: $f(n)=\begin{cases} 1, & n=0 \\ 1, & n=1 \\ f(n-1)+f(n-2), & n\geq 2 \end{cases}$ Біноміальні коефіцієнти $C_n^k=\frac{n!}{k!(n-k)!}$ розраховуються за рекурентним співвідношенням: $C_n^k=\begin{cases} C_{n-1}^{k-1}+C_{n-1}^k, & n>0 \\ 1, & k=n\geq 0\ або\ k=0, n<0 \\ 0, & в\ інших\ випадках \end{cases}$ Довести, що для перших чисел Фібоначчі справедлива формула: $F_{n+1}=\sum_{k=0}^{n/2} C_{n-k}^k$. Реалізувати рекурсивний варіант розв'язку задачі. Визначити глибину рекурсії.
	18.2. Увести з клавіатури натуральне число *n*. Вивести всі числа від 1 до *n*, використавши рекурсію. Контрольний тест: введено число 5, отриманий результат: 1 2 3 4 5
19.	19.1. Ввести ціле число *n* в десятковій системі числення з клавіатури. Перевести його у двійкову систему. Знайти кількість одиниць в двійковому представленні числа *n*, використовуючи рекурентне означення функції *f(n)*, де символ & означує операцію побітового логічного множення: $f(n)=\begin{cases} 0, & n=0 \\ 1+f(n\,\&\,(n-1)), & n\neq 0 \end{cases}$ Реалізувати рекурсивний варіант розв'язку задачі. Визначити глибину рекурсії.
	19.2. Потрібно сплатити поштове відправлення, вартість котрого складає *m* копійок, а в наявності тільки поштові марки номіналом *x, y, z* копійок. Скількома різними способами можна сплатити поштове відправлення? Розробити рекурсивну функцію для обчислення кількості зображень числа *m* у вигляді суми певних фіксованих чисел з використанням рекурентних співвідношень. Використати рекурентне співвідношення для чисел Фібоначчі.
20.	20.1.Увести з клавіатури ціле число *n* та дійсне *x*. Обчислити значення поліному від *x* степені *n* за рекурентним співвідношенням: $S_n(x)=\begin{cases} 0, & n=0 \\ 2x, & n=1 \\ \frac{2n}{n-1}S_{n-1}(x)+\frac{n-1}{2n}S_{n-2}(x), & n>1 \end{cases}$ Реалізувати рекурсивний варіант розв'язку задачі. Визначити глибину рекурсії.
	20.2. Використовуючи рекурсивні функції, перевірити, чи можна задане з клавіатури натуральне число представити у вигляді: а) добутку двох простих чисел; б) добутку трьох простих чисел; в) квадрата будь-якого простого числа; г) куба будь-якого простого числа.

Лабораторна робота 2.
Рекурентні співвідношення для тригонометричних, експоненціальних функцій та ланцюгові дроби

Мета

Опанувати теоретичні основи застосування рекурентних співвідношень для обчислення тригонометричних, експоненціальних, степеневих функцій та розробити програми функціональними мовам програмування для обчислення їх значень

Рейтингова таблиця лабораторної роботи 2

Вид роботи	**Кількість балів**	Deadline
Код задачі 1	2	жовтень
Код задачі 1	2	
Звіт	1	
Якість		
Разом	**5**	

Теоретичні відомості

Формула, що виражає член послідовності через один або декілька попередніх, називається *рекурентним співвідношенням*. Послідовність, члени якої задовольняють деякому рекурентному співвідношенню, називається рекурентною.

У загальному випадку рекурентне співвідношення визначає залежність члена послідовності $\{S_n\}$ від k попередніх членів: $S_n = F(S_{n-k},\ldots,S_{n-1})$.

Наближене значення суми ряду можна отримати або обмежуючись сумою перших n його членів, або обчислюючи суму з наперед заданою точністю. Формула загального члена даного ряду є достатньо простою, але використовувати її не раціонально, оскільки для кожного члена ряду треба обчислювати степінь і факторіал. Набагато вищої ефективності можна досягти, обчислюючи член ряду за допомогою рекурентного співвідношення. Найпростішими прикладами рекурентних послідовностей є арифметична та геометрична прогресії, елементи яких пов'язані з попередніми елементами співвідношеннями $a_n = a_{n-1}+d$ та $a_n = a_{n-1}\times q$, де d та q — деякі сталі величини, a_n – значення елемента ряду на кроці n.

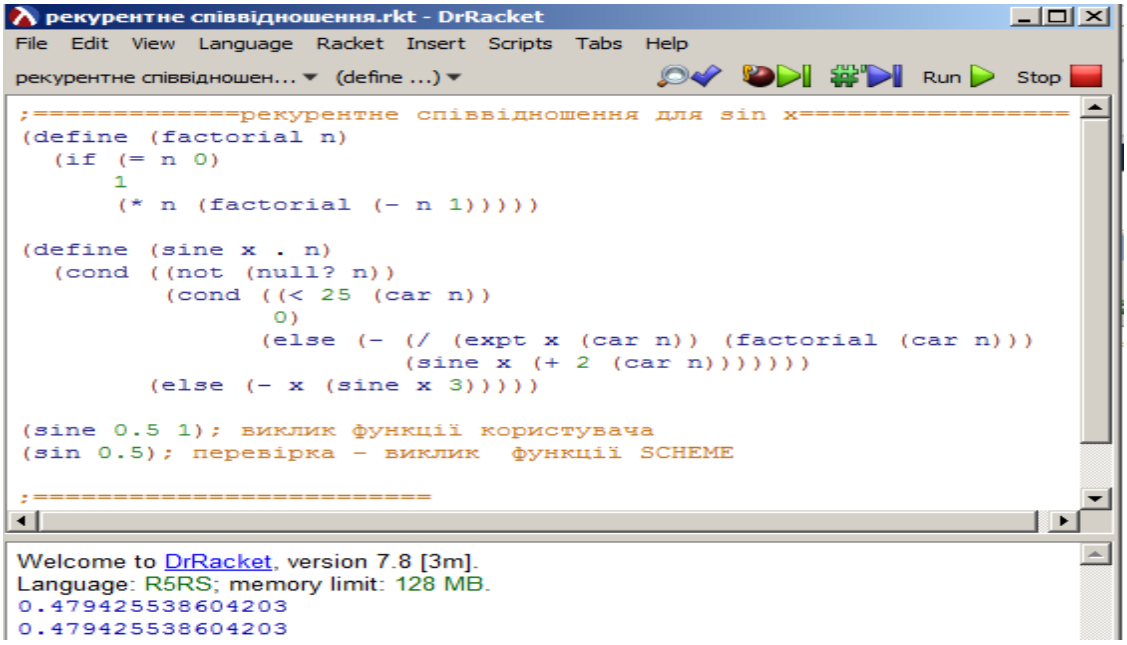
Із заданою точністю може бути обчислена сума лише збіжного ряду, а довільний степеневий ряд має певну область збіжності (можливо, порожню), тобто збігається не за всіх, а лише за деяких значень x (ряд, що розглядається нами як приклад, збігається для будь-якого дійсного x). По-друге, простий спосіб перевірки точності часткової суми ряду існує не для всіх рядів. Такий спосіб існує, зокрема, для знакозмінних рядів, абсолютні величини членів яких, починаючи з деякого номера, утворюють монотонно спадну послідовність. Для таких рядів сума всіх членів, починаючи від $(n+1)$-го, є меншою за модулем від n-го: $|a_n| > \sum_{i=n+1}^{\infty} a_i$.

Рекурентні формули розвинення функцій у ряди

Функція	**Розвинення у ряд Маклорена (Тейлора)**
sin x	$x - x^3/3! + x^5/5! - x^7/7! + \ldots$
cos x	$1 - x^2/2! + x^4/4! - x^6/6! + \ldots$
arcsin x	$x + \frac{x^3}{(2*3)} + \frac{(1*3)\,x^5}{(2*4*5)} + \frac{(1*3*5)\,x^7}{(2*4*6*7)} + \frac{(1*3*5*7)\,x^9}{(2*4*6*8*9)} + \ldots$
arctg x	$x - x^3/3 + x^5/5 - x^7/7 + \ldots$
sh x	$x + x^3/3! + x^5/5! + x^7/7! + \ldots$

ch x	$1 + x^2/2! + x^4/4! + x^6/6!+\ldots$
ln x	$(x–1) – (x–1)^2/2+(x–1)^3/3– (–1)^4/4+\ldots.$
e^x	$1 + x/1! + x^2/2! + x^3/3!+\ldots$
e^{-x}	$1 – x/1! + x^2/2! – x^3/3!+\ldots$
√x	$y_0 = 1, y_{n+1} = 1/2*(y_n+x/y_n), n = 0,1,2,\ldots$

Приклад коду



Зміст звіту

1. Титульний лист
2. Мета роботи
3. Умова завдання
4. Аналіз задачі та математичні забезпечення для розв’язання (у випадку математичної задачі)
5. Обґрунтування вибору мови функціонального програмування та IDE
6. Код
7. Скріншот роботи та результату
8. Оцінка достовірності результату (перевірка правильності результату)
9. Висновок

Завдання до лабораторної роботи 2

1. Написати процедури, що обчислюють задану функцію за допомогою рекурентних послідовностей, розвинувши її у ряд Маклорена (абоТейлора).
2. Параметр функції має змінюватися від заданого в процесі виклику мінімального значення до максимального значення із певним кроком.
3. Розвинення функції в ряд здійснювати із заданою точністю. Точність розрахунку задавати в діапазоні від 10^{-2} до 10^{-6}.
4. Для розвинення функції у ряд Маклорена (абоТейлора) створити власну функцію, яка розраховує суму ряду за рекурентним співвідношенням.
5. Значення функції tg(x) обчислювати через функції sin(x) та cos(x).
6. Визначити похибку обчислення наближених значень функції як різницю абсолютних значень наближеного обчислення та стандартного значення функції.

7. Стандартне значення функції обчислювати за допомогою бібліотечних математичних функцій.

Номер варіанта	Умова завдань
1	1.1 Обчислити значення функції y, розвинувши функцію cos(x) у ряд Тейлора. Аргумент x змінюється від -2 до 2 з кроком 0.5. Визначити похибку. $y=\begin{cases}\cos(x/2)/\cos(x^2), & -1\le x\le 0,\\ \cos^2(x/2)*\cos(2*x), & x>0.\end{cases}$
	1.2. Обчислити нескінчений ланцюговий дріб, задавши значення *n* при виклику функції $1+\cfrac{1}{3+\cfrac{1}{\cdots\ 2n-1+\cfrac{1}{2n+1}}}$
2	2.1. Обчислити значення функції y, розвинувши функцію у ряд Тейлора. Аргумент x змінюється від -2 до 2 з кроком 0.5. Визначити похибку. $y=\begin{cases}\sqrt{15-x^2}, & 1\le x\le 2,\\ \dfrac{1}{\sqrt{x+x^2}}, & -1\le x<1.\end{cases}$
	2.2. Обчислити скінчений ланцюговий дріб, задавши значення *a* при виклику функції $\cfrac{1}{a^2+\cfrac{2}{a^2+\cfrac{4}{a^2+\cfrac{8}{\cdots\ a^2+\cfrac{256}{a^2}}}}}$
3	Обчислити значення функції y, розвинувши функцію e^{-x} у ряд Тейлора. Аргумент x змінюється від -2 до 2 з кроком 0.5. Визначити похибку. $y=\begin{cases}e^{-x}+e^{-2x}, & 0\le x\le 2,\\ \dfrac{1}{e^{-(x+5)}}, & x>2.\end{cases}$
	Обчислити нескінчений ланцюговий дріб, задавши значення точності при виклику функції. Ланцюговий дріб виражає число π. $3+\cfrac{1^2}{6+\cfrac{3^2}{6+\cfrac{5^2}{6+\cfrac{7^2}{6+\cfrac{9^2}{6+\ \cdots}}}}}$
4	Обчислити значення функції y, розвинувши функцію sin(x) у ряд Тейлора. Аргумент x змінюється від -2 до 2 з кроком 0.5. Визначити $y=\begin{cases}\sin(x)-\sin(x/2), & 0<x<1,\\ \sin(x)^3-\sin(x+0125), & -2\le x\le 0.\end{cases}$
	Обчислити нескінчений ланцюговий дріб, задавши значення точності при виклику функції $\cfrac{4}{1+\cfrac{1^2}{3+\cfrac{2^2}{5+\cfrac{3^2}{7+\cfrac{4^2}{9+\ \cdots}}}}}$

5	5.1.Обчислити значення функції y, розвинувши функцію ln(x) у ряд Тейлора. Аргумент x змінюється від -1 до 3 з кроком 0.5. Визначити похибку $y=\begin{cases}\ln(x)+\ln(x/2), & 0<x<2,\\ \ln(x/2-1), & x\geq 2\end{cases}$
	Обчислити нескінчений ланцюговий дріб, задавши значення *точності* при виклику функції. $1+\cfrac{1}{2+\cfrac{1}{2+\cfrac{1}{2+\cfrac{1}{2+\ldots}}}}$
6	6.1. Обчислити значення функції y, розвинувши функцію ln(x) у ряд Тейлора. Аргумент x змінюється від -2 до 2 з кроком 0.5. Визначити похибку $y=\begin{cases}\ln(2*x)*\ln(x), & -2<x\leq 0,\\ \ln(x/2)-1, & 0<x\leq 1.\end{cases}$
	6.2. Обчислити нескінчений ланцюговий дріб, задавши значення *точності* при виклику функції $1+\cfrac{1}{1+\cfrac{1}{2+\cfrac{1}{1+\cfrac{1}{2+\ldots}}}}$
7	7.1. Обчислити значення функції y, розвинувши функцію cos(x) у ряд Тейлора. Аргумент x змінюється від -2 до 2 з кроком 0.5. Визначити похибку $y=\begin{cases}\cos^2(x)-\cos(x), & 0<x\leq 1,\\ \cos(x)^3+\cos(2*x), & -2\leq x\leq 0.\end{cases}$
	7.2. Обчислити нескінчений ланцюговий дріб, задавши значення *точності* при виклику функції $2+\cfrac{1}{4+\cfrac{1}{4+\cfrac{1}{4+\cfrac{1}{4+\ldots}}}}$
8	8.1. Обчислити значення функції y, розвинувши функцію у ряд Тейлора. Аргумент x змінюється від -2 до 2 з кроком 0.5. Визначити похибку. $y=\begin{cases}\sqrt{x-1}+\sqrt{x+1}, & -2<x\leq 0,\\ 1/\sqrt{x^2-1}, & 0<x<2.\end{cases}$
	8.2. Обчислити нескінчений ланцюговий дріб, задавши значення *точності* при виклику функції $1+\cfrac{1}{2+\cfrac{1}{1+\cfrac{1}{2+\cfrac{1}{1+\cfrac{1}{2+\ldots}}}}}$

9	9.1. Обчислити значення функції y, розвинувши функцію tg(x) у ряд Тейлора подавши її через sin(x) та cos(x).. Аргумент x змінюється від -3 до 3 з кроком 0.5. Визначити похибку $y=\begin{cases}(\mathrm{tg}(x)-(x/2))/\mathrm{tg}(2*x), & -2\le x<0,\\ \mathrm{tg}(x+2)-\mathrm{tg}^2(x), & 0<x\le 2.\end{cases}$
	9.2. Обчислити для заданого натурального числа *n* вираз: $\sqrt{3+\sqrt{6+\ldots+\sqrt{3(n-1)+\sqrt{3n}}}}$
10	10.1. Обчислити значення функції y, розвинувши функцію ln(1+x) у ряд Тейлора. Аргумент x змінюється від -3 до 3 з кроком 0.5. Визначити похибку $y=\begin{cases}\ln(1+x)^2+\ln(2*(1+x)), & -2\le x\le 0\\ \ln(1+x), & 0\le x\le 3\end{cases}$
	10.2. Обчислити для заданого натурального числа *n* вираз: $\sqrt{2+\sqrt{4+\ldots+\sqrt{2(n-1)+\sqrt{2n}}}}$
11	11.1Обчислити значення функції y, розвинувши функцію cos(x) у ряд Тейлора. Аргумент x змінюється від -2 до 3 з кроком 0.5. Визначити похибку $y=\begin{cases}\cos^2(x)*\cos(x^2), & -1\le x\le 1,\\ \cos(x/2)/\cos(x+0.5), & 1<x\le 2\end{cases}$
	11.2. Ввести з клавіатури натуральне число *n* . Необхідно отримати всі досконалі числа, менші за *n* . Досконалим називається число, значення якого дорівнює сумі всіх його дільників.
12	12.1Обчислити значення функції y, розвинувши функцію e^{-x} у ряд Тейлора. Аргумент x змінюється від -2 до 2 з кроком 0.5. Визначити похибку. $y=\begin{cases}e^{-x}+e^{-2x}, & -2\le x\le 0,\\ e^{-(x+5)}-e^{-x/2}, & 0<x<2.\end{cases}$
	12.2 Ввести з клавіатури натуральне число *n* . Визначити кількість цифр у числі *n* , підрахувати їх суму та знайти першу і останню цифри числа. Подання числа у вигляді структурованого типу (масивом або рядком) не використовувати .
13	13.1. Обчислити значення функції y, розвинувши функцію ln(x) у ряд Тейлора. Аргумент x змінюється від -2 до 4 з кроком 0.5. Визначити похибку. $y=\begin{cases}\dfrac{\ln(x)}{\ln(x-2)}, & 0<x<2,\\ \ln(x/2), & 2\le x\le 4.\end{cases}$
	13.2. Ввести з клавіатури два натуральних числа *m>100, n> m* . Визначити кількість чисел між *m, n*, які складаються з непарних цифр, або мають різні цифри. Подання числа у вигляді структурованого типу (масивом або рядком) не використовувати .
14	14.1. Обчислити значення функції y, розвинувши функцію у ряд Тейлора. Аргумент x змінюється від -2 до 2 з кроком 0.5. Визначити похибку. $y=\begin{cases}\sqrt{15-x^2}, & 1\le x\le 2,\\ \dfrac{1}{\sqrt{x+x^2}}, & -1\le x<1.\end{cases}$

	14.2. У трьох видах спорту стартує ***n*** осіб. Визначити, скільки існує можливих результатів, якими можуть закінчитися змагання, якщо кожна людина стартує в одному, довільно обраному виді спорту? Під результатом змагання розуміється розподіл місць для всіх спортсменів, що стартують в кожному виді.
15	15.1. Обчислити значення функції y, розвинувши функцію sin(x) у ряд Тейлора. Аргумент x змінюється від -2 до 2 з кроком 0.5. Визначити $y=\begin{cases}\sin(x)-\sin(x/2), & 0<x<1,\\ \sin(x)^3-\sin(x+0125), & -2\le x\le 0.\end{cases}$
	15.2. Учасники лижних змагань стартують з інтервалом 30 секунд. Щоб визначити порядок старту, спортсмени тягнуть жереб, який вказує номер старту. Визначити кількість різних послідовностей виходу лижників на старт, якщо в змаганнях брало участь *n* осіб. Через який проміжок часу всі спортсмени будуть на лижах? Значення кількості спортсменів *n* вводити з клавіатури.
16	16.1. Обчислити значення функції y, розвинувши функцію arctg(x) у ряд Тейлора. Аргумент x змінюється від 0 до 3 з кроком 0.5. Визначити похибку $y=\begin{cases}\text{arctg}(x)/\text{arctg}(x-5), & x\ge 1,\\ \text{arctg}(x)+\text{arctg}(2x), & 0\le x<1\end{cases}$
	16.2. Два натуральних числа називаються "дружніми", якщо кожне з них дорівнює сумі всіх дільників іншого, крім самого цього числа. Визначити всі пари "дружніх" чисел, що лежать у діапазоні [200, 300] .
17	17.1. Обчислити значення функції y, розвинувши функцію e^x у ряд Тейлора. Аргумент x змінюється від -2 до 2 з кроком 0.5. Визначити похибку. $y=\begin{cases}e^{\text{arctg}(x)}, & -2\le x<0,\\ e^{x^2}+1, & 0\le x\le 1.\end{cases}$
	17.2. Натуральне число із *n* цифр є числом Армстронга, якщо сума його цифр, піднесених до *n*-го степенюя, дорівнює самому числу (наприклад, 153 = 1^3 + 5^3 + 3^3=1+125+27). Визначити всі числа Армстронга, що складаються з двох, трьох та чотирьох цифр
18	18.1. Обчислити значення функції y, розвинувши функцію sin(x) у ряд Тейлора. Аргумент x змінюється від -2 до 2 з кроком 0.5. Визначити похибку $y=\begin{cases}\sin^2(x)-\sin(x), & 0<x\le 1,\\ \sin(x)^3+\sin(2*x), & -2\le x\le 0.\end{cases}$
	18.2. Для заданого натурального числа *n* визначити всі Піфагорові трійки натуральних чисел, кожне з яких не перевищує *n* . Піфагорові трійки натуральних чисел *a, b, c* відповідають умові: *a^2+b^2=c^2,* (*a<=b<=c<=n*) .
19	19.1. Обчислити значення функції y, розвинувши функцію e-x у ряд Тейлора. Аргумент x змінюється від -2 до 2 з кроком 0.5. Визначити похибку. $y=\begin{cases}e^{-(x+2)}+e^{-x/3}, & -2\le x\le 0,\\ e^{-(x-5)}-e^{-x^2}, & 0<x<2.\end{cases}$
	19.2. Ввести з клавіатури два натуральних числа *m, n*. Визначити кількість способів, якими можна розсадити *n* людей серед *m* людей за круглим столом. Розміщення, що відрізняються тільки циклічним переміщенням навколо столу, вважаємо однаковими.

20	20.1. Обчислити значення функції y, розвинувши функцію sh(x) у ряд Тейлора. Аргумент x змінюється від -3 до 3 з кроком 0.5. Визначити похибку. $y = \begin{cases} \mathrm{sh}(x) + \mathrm{sh}(2 + x), & -2 \le x \le 0, \\ \mathrm{sh}^2(x) / \mathrm{sh}(x^3), & 1 < x \le 2. \end{cases}$
	20.2. Обчислити скінчений ланцюговий дріб, задавши значення *n* при виклику функції $\cfrac{1}{3+\cfrac{1}{4+\cfrac{}{..\cfrac{}{...\cfrac{1}{1991}}}}}$

Лабораторна робота 3.
Форми lambda та let, вираз присвоєння set! для розв'язання нелінійних рівнянь та чисельного інтегрування функцій

Мета

Розв'язати нелінійні рівняння та здійснити чисельне інтегрування функцій наближеними методами, використовуючи мови функціонального програмування та lambda, let та set! форми.

Рейтингова таблиця лабораторної роботи 3

Вид роботи	Кількість балів	Deadline
Код задачі 1	2	жовтень
Код задачі 1	2	
Звіт	1	
Якість		
Разом	**5**	

Теоретичні відомості

Для визначення нової функції мовою Scheme використовується така форма:

```
(lambda (<arguments>) (<body> )),
```

де **<arguments>** - список аргументів (через пробіл); **<body>** - правильний S-вираз, який набуває значення.

Такий запис функцій називається лямбда-функцією, оскільки функція немає імені для її зручного виклику. Наприклад, визначемо функцію одного аргументу, яка додає до переданого аргументу число 10:

(lambda (x) (+ x 10))

Згідно концепції Lisp, всі функції викликаються у форматі

```
(function_name param1 param2 … paramN)
```

Якщо функція є без імені, то можна підставити замість **function_name** сам опис функції. Наприклад, виклик вищеописаної функції для параметру 20:

((lambda (x)(+ x 10)) 20)

Недоліком такого опису функції є складність її виклику, оскільки для цього потрібно завжди використовувати повний опис функції.

Для створення функції з іменем потрібно використати один із форматів опису функції:

```
(define <name> (lambda (<arguments>) (<body>)))
```

де **<name>** - ім'я функції; **<arguments>** - список аргументів (через пробіл); **<body>** - правильний S-вираз, який набуває значення.

Приклад.

```
((lambda(x)
      (+ x 42))
 23)
```

Правилами для ключового слова **lambda** визначено, що перша підформа є списком параметрів, а решта підформи - тілом процедури.

Змінні Scheme, прив'язані визначеннями або виразами **let** або **lambda**, фактично прив'язуються не безпосереднє до об'єктів, що визначені у відповідних прив'язках, а до адреси пам'яті, що містить ці об'єкти. Вміст за цими адресами згодом може бути змінено за допомогою присвоювання:

```
;example1

(let ((x 23))
    (set! x 42)

;example2

(let ((a 2)
        (b 3)
    (+ a b))
```

У даному випадку тіло виразу **let** складається з двох обчислюваних послідовно виразів зі значенням остаточного виразу, що приймає значення всього виразу **let**. Вираз (**set**! x 42) є присвоєнням, що вказує "замінити об'єкт за адресою, на яку вказує x, на 42 ". Таким чином, попереднє значення x, що дорівнює 23, змінюється на 42.

Методи розв'язання нелінійних рівнянь

Метод ділення відрізка навпіл (метод бісекції)

Для знаходження кореня рівняння $f(x)=0$, що належить відрізку $[a, b]$, ділимо цей відрізок навпіл. Якщо $f\left(\frac{a+b}{2}\right)=0$, то $\chi=\frac{a+b}{2}$ є коренем рівняння. Якщо $f\left(\frac{a+b}{2}\right)\neq 0$, то вибираємо ту з половин $\left[a,\frac{a+b}{2}\right]$ або $\left[\frac{a+b}{2},b\right]$, на кінцях якої функція $f(x)$ має протилежні знаки. Новий звужений відрізок $[a_1,b_1]$ знову ділимо навпіл і робимо ті самі дії.

Метод хорд

При розв'язанні нелінійного рівняння $f(x)=0$ методом хорд задаються відрізок $[a, b]$, на якому існує тільки один розв'язок, і точність ε . На 0 -й ітерації методу $[a_0, b_0]$ $[a, b]$, на k -й ітерації методу маємо поточний відрізок $[a_k, b_k]$. Потім через дві точки з координатами $(a_k , f(a_k))$ й $[b_k, f(b_k)]$ проводимо відрізок прямої лінії (хорду) і визначаємо точку перетину цієї лінії з віссю абсцис (точка x_k). Якщо при цьому $f(a_k)\times f(x_k) < 0$, то праву межу інтервалу переносимо в точку x_k (тобто $b_{k+1} = x_k$, $a_{k+1} = a_k$). Якщо зазначена умова не виконується, то в точку x_k переноситься ліва межа інтервалу ($a_{k+1} = x_k$, $b_{k+1} = b_k$). Пошук розв'язку припиняється при досягненні заданої точності, тобто $f(x_k)$ ( ) < ε. Точка перетину хорди з віссю абсцис визначається за формулою:

$$x_{k+1} = a_k + \left|\frac{f(a_k)}{f(b_k)-f(a_k)}\times(b_k-a_k)\right.$$

Метод дотичних (метод Ньютона).

При розв'язанні нелінійного рівняння $f(x)=0$ методом дотичних задаються початкове наближення x_0 і точність ε. Потім у точці $(x_0, f(x_0))$ проводиться дотична до графіка $f(x)$ й визначається точка x_1 перетину дотичної з віссю абсцис. У точці(x_1,$f(x_1)$) знову будується дотична, обчислюється наступне наближення шуканого розв'язку x_2 і т. д. Зазначена процедура повторюється доти, поки $|f(x_i)|>$ ε . Точка перетину $(k+1)$-ої дотичної з віссю абсцис визначається за формулою:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Умова збіжності методу дотичних $f(x_0)\times f'(x_0)>0$

Метод простої ітерації

Метод простої ітерації полягає в тому, що рівняння $f(x)=0$ попередньо приводиться до канонічного вигляду $x = \phi(x)$. Ітерації виконуються за правилом $x_{k+1}=\phi(x_k)$ для $k=0,1,\ldots$ Зазначена процедура припиняється при досягненні заданої точності, тобто $|f(x_i)| = |x_k - \phi(x_k)|\leq \varepsilon$. Умова збіжності методу ітерацій:$|\phi'(x)| \leq q < 1$ для всіх $x \in [a,b]$.

Формула Сімпсона

Якщо для кожної пари відрізків $[x_i, x_{i+2}]$ побудувати многочлен другого ступеня, потім проінтегрувати його і скористатися властивістю адитивності інтеграла, то одержимо формулу Сімпсона.

$$\int_{x_0}^{x_0+2h} f(x)dx \approx \frac{h}{3}[f(x_0) + 4f(x_0 + h) + f(x_0 + 2h)]$$

Отримане для інтеграла $\int_{x_0}^{x_2} \varphi(x)dx$ значення збігається із площею криволінійної трапеції, обмеженою віссю x, прямими $x = x_0, x = x_2$ і параболою, що проходить через точки $(x_0, f(x_0)),(x_1, f(x_1)),(x_2, f(x_2))$.

Формули прямокутників

Формула лівих прямокутників

$$\int_a^b f(x)\,dx \approx \frac{b-a}{n}(y_0 + y_1 + \ldots + y_{n-1}) = \frac{b-a}{n}\sum_{i=0}^{n-1} f(x_i)$$

Формула правих прямокутників

$$\int_a^b f(x)\,dx \approx \frac{b-a}{n}(y_1 + y_2 + \ldots + y_n) = \frac{b-a}{n}\sum_{i=1}^{n} f(x_i)$$

Формула середніх прямокутників

$$\int_a^b f(x)\,dx \approx \frac{b-a}{n}\sum_{i=1}^{n} f\left(\frac{x_{i-1} + x_i}{2}\right)$$

Формула трапецій

$$\int_a^b f(x)\,dx \approx \frac{b-a}{n}\left(\frac{y_0 + y_n}{2} + y_1 + y_2 + \ldots + y_{n-1}\right) = \frac{b-a}{2n}\sum_{i=0}^{n} f(x_i)$$

Приклад коду

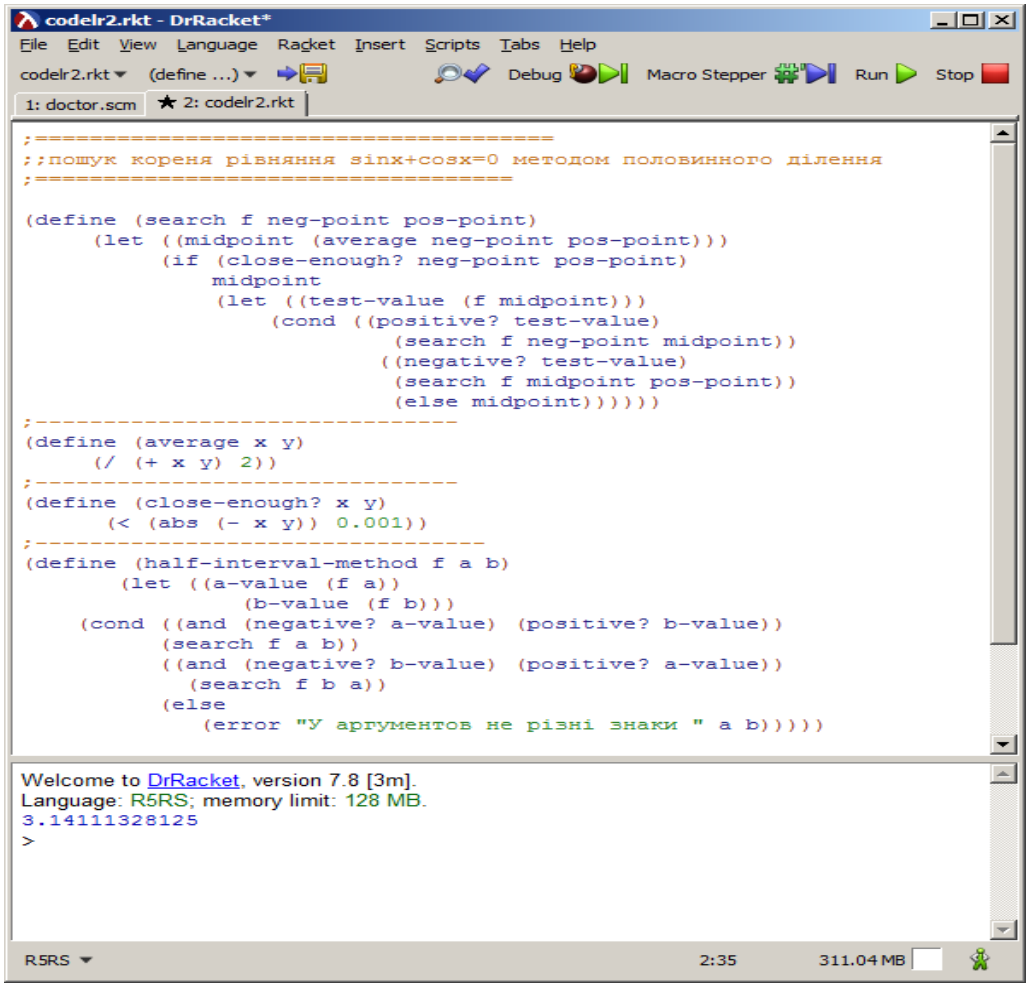


Рис. 1. Приклад коду задачі пошук кореня рівнянн методом половинного ділення

Завдання до лабораторної роботи 3

1. **Написати процедури, що знаходять корені нелінійних рівнянь, використовуючи форми lambda, let, set!**
2. **Написати процедури, що обчислюють інтеграл функції за формулами прямокутників, трапецій, Сімпсона (парабол)**

Номер варіанта	Умова завдань
1.	1.1 Розв'язати нелінійне рівняння $x=\cos(x)$ методами перебору та хорд, визначивши інтервал $[a, b]$, на якому існує рішення рівняння. Значення a, b інтервалу вибрати самостійно. Порівняти результати розв'язків двома методами.
	1.2. За допомогою формули Сімпсона інтеграл функції $f(x)$ від a до b наближено обчислюється у вигляді: $$\frac{h}{3}[y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_4 + \ldots + 2y_{n-2} + 4y_{n-1} + y_n]$$ де $h = (b - a) / n$, для якогось парного цілого числа n, $y_k = f(a + kh)$. (Збільшення n підвищує точність наближеного обчислення.) Визначити процедуру, яка приймає в якості аргументів f, a, b, n, та повертає значення інтеграла, обчисленого за формулою Сімпсона. $\int_0^{\pi} \frac{\cos x}{\sqrt{1-x^2}}\,dx$

2.	2.1 Знайти корені нелінійного рівняння виду $x = \ln(x)+2$. Пошук наближеного значення хоча б одного кореня рівняння $f(x) = 0$ на відрізку $[a; b]$ здійснювати методами перебору та дотичних. Значення a, b інтервалу вибрати самостійно. Порівняти результати розв'язків двома методами.
	2.2 Якщо *f(x)* - функція, а *dx* - деяке мале число, то згладжена версія *f(x)* є функція, значення якої в точці *x* є середнє між *f (x - dx), f (x)* і *f (x + dx)*. Напишіть процедуру, яка в якості параметрів приймає процедуру, яка обчислює *f(x)*, і повертає процедуру, яка обчислює згладжену версію *f(x)*.
3.	3.1. Знайти корені нелінійного рівняння виду $\boldsymbol{x^2=e^{-x}}$. Пошук наближеного значення хоча б одного кореня рівняння $\boldsymbol{f(x) = 0}$ на відрізку $[a; b]$ здійснювати методами перебору та хорд. Значення *a, b* інтервалу вибрати самостійно. Порівняти результати розв'язків двома методами.
	3.2 Написати процедури для обчислити інтеграла за формулами прямокутників і трапецій. Порівняти результати обчислення. $\int_0^1 \frac{dx}{1+x^2}$
4.	4.1 Знайти корені нелінійного рівняння виду $x = 3e^{-3x}$. Пошук наближеного значення хоча б одного кореня рівняння $f(x) = 0$ на відрізку $[a; b]$ здійснювати методами дотичних та Ньютона. Значення a, b інтервалу вибрати самостійно. Порівняти результати розв'язків двома методами.
	4.2 Написати процедури для обчислити інтеграла за формулами прямокутників і Сімпсона. Порівняти результати обчислення. $\int_0^1 \cos(x^2 + x + 1)dx$
5.	5.1 Знайти корені нелінійного рівняння виду $\cos(x) - x + 5=0$. Пошук наближеного значення хоча б одного кореня рівняння $f(x) = 0$ на відрізку $[a; b]$ здійснювати методами бісекції та Ньютона. Значення a, b інтервалу вибрати самостійно. Порівняти результати розв'язків двома методами.
	5.2 Написати процедури для обчислити інтеграла за формулами Сімпсона і трапецій. Порівняти результати обчислення. $\int_0^1 \sin(x^2 + x + 1)dx$
6.	6.1. Знайти корені нелінійного рівняння виду $x^2 + 4\sin x = 0$. Пошук наближеного значення хоча б одного кореня рівняння $f(x) = 0$ на відрізку $[a; b]$ здійснювати методами хорд та простої ітерації. Значення a, b інтервалу вибрати самостійно. Порівняти результати розв'язків двома методами.
	6.2 Написати процедури для обчислити інтеграла за формулами прямокутників і Сімпсона. Порівняти результати обчислення. $\int_{-0.5}^{1.3} \frac{x^2}{\sqrt{x^2+1}} dx$
7.	7.1. Знайти корені нелінійного рівняння виду $x^2 - 2x + \ln(x) = 0$. Пошук наближеного значення хоча б одного кореня рівняння $f(x) = 0$ на відрізку $[a; b]$ здійснювати методами Ньютона та простої ітерації. Значення a, b інтервалу вибрати самостійно. Порівняти результати розв'язків двома методами. .

	7.2 Написати процедури для обчислити інтеграла за формулами лівих і правих прямокутників. Порівняти результати обчислення. $\int_{0.4}^{1.8} \frac{x^2+1.4}{\sqrt{x^2+0.2}}\,dx$
8.	8.1 Знайти корені нелінійного рівняння виду $x^2 - 2\cos x - 1 = 0$. Пошук наближеного значення хоча б одного кореня рівняння $f(x) = 0$ на відрізку $[a; b]$ здійснювати методами дотичних та бісекції. Значення a, b інтервалу вибрати самостійно. Порівняти результати розв'язків двома методами.
	8.2. Написати процедури для обчислити інтеграла за формулами лівих прямокутників і трапеції. Порівняти результати обчислення. $\int_{0}^{\pi} \cos(x - \sin x)\,dx$
9.	9.1. Знайти корені нелінійного рівняння виду $2x - \cos x = 0$. Пошук наближеного значення хоча б одного кореня рівняння $f(x) = 0$ на відрізку $[a; b]$ здійснювати методами простої ітерації та дотичних. Значення a, b інтервалу вибрати самостійно. Порівняти результати розв'язків двома методами.
	9.2. Написати процедури для обчислити інтеграла за формулами правих прямокутників і Сімпсона. Порівняти результати обчислення. $\int_{0}^{\pi} \sin(x - \cos x)\,dx$
10.	10.1. Знайти корені нелінійного рівняння виду $x^3 - 4x + 6 = 0$. Пошук наближеного значення хоча б одного кореня рівняння $f(x) = 0$ на відрізку $[a; b]$ здійснювати методами Ньютона та перебором. Значення a, b інтервалу вибрати самостійно. Порівняти результати розв'язків двома методами.
	10.2 Написати процедури для обчислити інтеграла за формулами середніх прямокутників і Сімпсона. Порівняти результати обчислення. $\int_{0}^{\pi} \sin(2\cos x)\,dx$
11.	11.1. Знайти корені нелінійного рівняння виду $x^2 - \cos x = 0$. Пошук наближеного значення хоча б одного кореня рівняння $f(x) = 0$ на відрізку $[a; b]$ здійснювати методами простої ітерації та перебором. Значення a, b інтервалу вибрати самостійно. Порівняти результати розв'язків двома методами.
	11.2 Написати процедури для обчислити інтеграла за формулами трапецій і Сімпсона. Порівняти результати обчислення. $\int_{0.1}^{2} \frac{\cos x}{\sqrt{x}}\,dx$
12.	12.1. Знайти корені нелінійного рівняння виду $\cos x - 3x = 0$. Пошук наближеного значення хоча б одного кореня рівняння $f(x) = 0$ на відрізку $[a; b]$ здійснювати методами простої ітерації та перебором. Значення a, b інтервалу вибрати самостійно. Порівняти результати розв'язків двома методами.
	12.2. Написати процедури для обчислити інтеграла за формулами трапецій і прямокутників. Порівняти результати обчислення. $\int_{0}^{\pi} \cos(2\sin x)\,dx$

13.	13.1 Знайти корені нелінійного рівняння виду $2 - x - \ln x = 0$. Пошук наближеного значення хоча б одного кореня рівняння $f(x) = 0$ на відрізку $[a; b]$ здійснювати методами Ньютона та перебором. Значення a, b інтервалу вибрати самостійно. Порівняти результати розв'язків двома методами.
	13.2 Написати процедури для обчислити інтеграла за формулами прямокутників і Сімпсона. Порівняти результати обчислення. $\int_0^{\pi} e^{\cos x} \cos 2x dx$
14.	14.1. Знайти корені нелінійного рівняння виду $e^x + \ln x - 10x = 0$. Пошук наближеного значення хоча б одного кореня рівняння $f(x) = 0$ на відрізку $[a; b]$ здійснювати методами простої ітерації та дотичних. Значення a, b інтервалу вибрати самостійно. Порівняти результати розв'язків двома методами.
	14.2. Написати процедури для обчислити інтеграла за формулами прямокутників і Сімпсона. Порівняти результати обчислення. $\int_{-1}^{\pi} xe^{-x} dx$
15.	15.1. Знайти корені нелінійного рівняння виду $e^x + 2x - 26 = 0$. Пошук наближеного значення хоча б одного кореня рівняння $f(x) = 0$ на відрізку $[a; b]$ здійснювати методами хорд та перебором. Значення a, b інтервалу вибрати самостійно. Порівняти результати розв'язків двома методами.
	15.2. Написати процедури для обчислити інтеграла за формулами прямокутників і трапецій. Порівняти результати обчислення. $\int_1^3 x^{-1} e^x dx$
16.	16.1. Знайти корені нелінійного рівняння виду $3x - \cos x - 1 = 0$. Пошук наближеного значення хоча б одного кореня рівняння $f(x) = 0$ на відрізку $[a; b]$ здійснювати методами простої ітерації та дотичних. Значення a, b інтервалу вибрати самостійно. Порівняти результати розв'язків двома методами.
	16.2. Написати процедури для обчислити інтеграла за формулами прямокутників і трапецій. Порівняти результати обчислення. $\int_0^{\pi} x^4 e^{-x^2} dx$
17.	17.1. Знайти корені нелінійного рівняння виду $x^3 - 3x^2 + 6x + 3 = 0$. Пошук наближеного значення хоча б одного кореня рівняння $f(x) = 0$ на відрізку $[a; b]$ здійснювати методами бісекції та Ньютона. Значення a, b інтервалу вибрати самостійно..
	17.2. Написати процедури для обчислити інтеграла за формулами Сімпсона і трапецій. Порівняти результати обчислення $\int_0^{\pi/2} e^{-\cos x} \cos(\sin x) dx$
18.	18.1 Знайти корені нелінійного рівняння виду $x^2 - \ln(1 + x) - 3 = 0$. Пошук наближеного значення хоча б одного кореня рівняння $f(x) = 0$ на відрізку $[a; b]$ здійснювати методами дотичних та Ньютона. Значення a, b інтервалу вибрати самостійно. Порівняти результати розв'язків двома методами.
	18.2. Написати процедури для обчислити інтеграла за формулами і трапецій. Порівняти результати обчислення. $\int_{0.5}^{1.6} \frac{x^2 + 0.5}{\sqrt{x^2 + 1}} dx$

19.	19.1. Знайти корені нелінійного рівняння виду $3^x - 5x + 2 = 0$. Пошук наближеного значення хоча б одного кореня рівняння $f(x) = 0$ на відрізку $[a; b]$ здійснювати методами перебором та бісекції. Значення a, b інтервалу вибрати самостійно. Порівняти результати розв'язків двома методами.
	19.2. Написати процедури для обчислити інтеграла за формулами прямокутників і трапецій. Порівняти результати обчислення. $\int_{1.2}^{2.8} \frac{\lg(1+x^2)}{2x-1}\,dx$
20.	20.1. Знайти корені нелінійного рівняння виду $x^2 - 1 - \cos 5x = 0$. Пошук наближеного значення хоча б одного кореня рівняння $f(x) = 0$ на відрізку $[a; b]$ здійснювати методами бісекціїї та дотичних. Значення a, b інтервалу вибрати самостійн Порівняти результати розв'язків двома методами.
	20.2. Написати процедури для обчислити інтеграла за формулами Сімпсона і трапецій. Порівняти результати обчислення. $\int_{0}^{\pi/2} e^{-\cos x}\cos(\sin x)\,dx$

Зміст звіту

1. Титульний лист
2. Мета роботи
3. Умова завдання
4. Аналіз задачі та математичні забезпечення для розв'язання (у випадку математичної задачі)
5. Обґрунтування вибору мови функціонального програмування та IDE
6. Код
7. Скріншот роботи та результату
8. Оцінка достовірності результату (перевірка правильності результату)
9. Висновок

Лабораторна робота 4
Програмування списків мовами функціонального програмування

Мета

Опанувати теоретичні основи використання списків функціональними мовами та розробити програми обробки списків

Рейтингова таблиця лабораторної роботи 4

Вид роботи	**Кількість балів**	Deadline
Код задачі 1	2	листопад
Код задачі 1	2	
Звіт	1	
Якість		
Разом	**5**	

Теоретичні відомості

Список. Послідовність зв'язаних між собою елементів, кожен з яких є або атомом, або списком, або парою називають у функціональному програмуванні списком (List). Списки позначаються в дужках, елементи списку розділяються пробілами.

Пара. Об'єднання двох елементів в одне ціле називається парою. Елементами пари можуть бути об'єкти допустимої у функціональному програмуванні структури, **тобто атом, пара, список**. Пара позначається в дужках, між елементами ставиться крапка, яка розділена з обох сторін пробілами.

Члени списку організовані в пам'яті комп'ютера у вигляді послідовностей комірок, розділених на дві частини. У першій частині комірки вказується інформація про член списку. Це може бути безпосереднє значення, якщо інформація подається атомом, або адреса комірки, де є перший елемент вкладеного об'єкту, в протилежному випадку. У другій частині комірки вказується адреса зв'язку, тобто адреса знаходження наступного члена заданого списку. Якщо член списку є останнім, то у другій частині комірки вказується символ **NIL** ("ніщо).

Порожній список позначається ( ).

Процедури обробки списків

Функція **car** повертає голову переданого в якості аргументу списку (перший елемент списку називається головою, а залишок списку - хвостом).

Функція **cdr** повертає хвостову частину списку.

Функція **nth** повертає елемент списку по його номеру. Ця функція отримує два аргументи: перший - номер елемента в переданому в якості аргументу списку (нумерація починається з нуля); другий - сам список.

Функції **first, second, third**, повертають відповідно перший, другий, третій, елементи переданого в якості аргументу списку.

Функція **last** повертає останній елемент списку.

Функція **cons** будує новий список з переданих їй як аргументи голови і хвоста списку.

Функція **list** створює список з переданих їй як елементи аргументів. Кількість аргументів довільна.

Стандартні процедури в мові Scheme

Процедура	**Семантика**
pair?	Предикат перевірки наявності пари
cons	Конструктор пари
car	Визначення голови списку
cdr	Визначення другого елемента пари або хвостової частини списку
set-car!	Замінити перше значення списку на нове значення

set-cdr!	Створити пару з першого та вказаного значень
null?	Предикат перевірки списку на пустоту
list?	Предикат перевірки наявності списку
list	Створення списку
length	Довжина списку
append	Додавання списку
reverse	Інвертування списку
list-tail	Хвостова частина після заданої позиції
list-ref	Значення елемента списку на заданій позиції
memq	Хвостова частина, починаючи з вказаного значення = member
memv	Хвостова частина, починаючи з вказаного значення = member
member	Хвостова частина, починаючи з вказаного значення
assq,	Повертає пару із списку. Перший елемент пари вказаний як аргумент
assv,	Повертає пару із списку. Перший елемент пари вказаний як аргумент
assoc	Повертає пару із списку. Перший елемент пари вказаний як аргумент
list->vector	Перетворити список на вектор
vector->list	Перетворити вектор на список
list->string	Перетворити список у рядок
string->list	Перетворити рядок на список

Комбінації селекторів

(cadr x) = (car (cdr x))
(cdar x) = (cdr (car x))
(caar x) = (car (car x))
(cddr x) = (cdr (cdr x))
(cadar x) = (car (cdr (car x)))

Приклади коду на **Racket**

```
;========== копія списку==========
(define (list-copy lis)
  (cond ((null? lis)
         '())
        (else
         (cons (car lis)
               (list-copy (cdr lis))))))
(list 1 2 3 4 5)              ; виклк процедур
(list-copy (list 1 2 3 4 5))
;=========додавання двох списків==========
(define (my-append lis1 lis2)
  (cond ((null? lis1)
         lis2)
        (else
         (cons (car lis1)
               (my-append (cdr lis1) lis2)))))
(define lis1 (list 1 2 3 4 5))                ; виклк процедур
(define lis2 (list 6 7 8 9 0))
(my-append lis1 lis2)
;==========інвертування списку=========================
(define (my-reverse lis)
  (if (null? lis)
      '()
      (append (my-reverse (cdr lis))
              (list (car lis)))))

(define lis1 (list 1 2 3 4 ))
(my-reverse lis1)
;==========кількість елементів списку=================
(define (my-length lis)
```

```
  (cond ((null? lis)
         0)
        (else
         (+ 1 (my-length (cdr lis))))))

(my-length lis1)
;==============доступ до елементів списку по номеру======================
(define (my-list-ref items n)
    (if (= n 1)
        (car items)
        (my-list-ref (cdr items) (- n 1 ))))
(define squares (list 1 4 9 16 25))
(my-list-ref squares 1)
(my-list-ref squares 2)
(my-list-ref squares 4)
;(my-list-ref squares 10) ;error
;================================
(display 'set-car)
(define a (list 1 2 3));?
(set-car! a 3);?
(display a)
(set-cdr! a 9);?

(define a (list 9 8 7));?
(set-cdr! a 1);?
(display a)
;==========доступ до елементів списку===============
(define l (list 1 2 3 4 5))
(list-tail l 2);{3 4 5}

(list-ref l 2)  ;3
(member 2 (list 1 2 3 4));(2 3 4)

(member 9 (list 1 2 3 4))   ;#f

(memv 2 (list 1 2 3 4));(2 3 4)
(memq 2 (list 1 2 3 4))
;========================
(assoc 7 (list (list 1 2) (list 3 4) (list 5 6)))
(assq 'c (list (list 'a 'b) (list 'c 'd) (list 'e 'f)))

;============================
 (define v (vector 'A 'p 'p 'l 'e))
(vector->list v)
(define lst(list 't 'o 'm 'a 't 'o))
(list->vector lst )
```

Зміст звіту

1. Титульний лист
2. Мета роботи
3. Умова завдання
4. Аналіз задачі та математичні забезпечення для розв’язання (у випадку математичної задачі)
5. Обґрунтування вибору мови функціонального програмування та IDE
6. Код
7. Скріншот роботи та результату
8. Оцінка достовірності результату (перевірка правильності результату)
9. Висновок

Завдання до лабораторної роботи 4

1. **Написати процедури, що створюють список, модифікують його, здійснюють пошук та упорядкування значень.**
2. **Написати процедури, що моделюють бізнес-процеси в компаніях та за допомогою різних сервісів. Бізнес-процеси подати у вигляді списків.**

Номер варіанта	Умова завдань
1.	1.1 Створити список натуральних чисел (натуральні числа >= 1), кратних 3, задавши їх кількість. Вивести створений список. Виконати такі операції: a) додати елементи в список на задану позицію в списку; b) підрахувати кількість парних елементів в списку; c) замінити усі парні значення списку на середнє арифметичне елементів списку.
	1.2. Написати код, що моделює роботу сортувального вузла на залізниці в процесі формування складу потягу. На сортувальному вузлі формуються потяги. Нехай існує два типи вагонів. На кожний напрямок потяг складається з вагонів одного типу. Відомі такі часові характеристики для кожного потягу: інтервали між потягами, кількість вагонів у потягу, тривалість причеплення вагону до потягу, тривалість огляду сформованого потягу, загальна тривалість простою потягу до його відправлення. Вивести на екран склад кожного потягу та сценарій виконання дій з урахування часових характеристик.
2.	2.1 Створити список парних натуральних чисел (натуральні числа >=1), задавши їх кількість. Вивести створений список. Виконати такі операції: a) Додати елементи на початок списку. Кількість доданих елементів задається; b) Здійснити пошук заданого елемента та визначення позиції його в списку, у випадку відсутності елемента вивести відповідне повідомлення; c) підрахувати кількість елементів, значення яких в заданому діапазоні.
	2.2 Написати код, що моделює роботу планувальника процесів в операційні системі. Нові процеси знаходяться у вхідній черзі і очікують звільнення ресурсу — адресного простору основної пам'яті. Готові до виконання процеси розташовуються в основній пам'яті і зв'язані чергою готових процесів. Процеси в цій черзі чекають на отримання процесорного часу. Процес у стані чекання завершення операції введення/виведення знаходиться в одній з черг до пристроїв уведення/виведення. Під час виконання вказівник процесу мігрує між різними чергами під управлінням програми-планувальник, яка вирішує, який із процесів, що знаходяться в черзі готових процесів, повинен бути переданий на виконання CPU. Вивести на екран черги процесів та сценарій їх міграції.
3.	3.1. Створити список непарних натуральних чисел (натуральні числа >=1), задавши їх кількість. Вивести створений список. Виконати такі операції: a) додати елементи в кінець списку. Кількість доданих елементів задається; b) інвертувати список із заданої позиції до кінця (до заданої позиції) списку; c) сформувати новий список із елементів, які є квадратами чисел вхідного списку.
	3.2 Написати код, що моделює процес здачі усних іспитів в ЗВО в режимі real time. Студенти, що допущені на іспит, сформовані у список. В процесі проходження іспиту екзаменатор викликає перших двох студентів зі списку, які отримують запитання. Поки один студент спілкується з екзаменатором, другий готовить письмову відповідь, Екзаменатор задає по два запитання, відповідь на які мають дати студенти. Після спілкування з двома студентами тривалістю m хвилин, до екзаменатора викликають наступні два студента. Якщо відповіді студентів не зараховуються, то формується новий список студентів на перездачу. Вивести на екран списки студентів, які здали іспит та список тих, кому призначена перездача. Студентів можна ідентифікувати порядковими номерами.
4.	4.1 Створити список факторіалів цілих чисел, задавши їх кількість. Вивести створений список. Виконати такі операції: a) Здійснити пошук максимального та мінімального елементів списку; b) додати нові елементи в список між максимальним і мінімальним. Кількість доданих елементів задавати; c) Визначити суму доданих елементів.

	4.2 Написати код, що моделює процес обслуговування пацієнтів у сімейного лікаря. Для доступу до лікаря створений список пацієнтів, які ідентифікуються порядковими номерами. До лікаря заходить пацієнт, який може перебувати у нього певний час. Якщо лікар підозрює у пацієнта коронавірус, він видає направлення в лабораторію і пацієнт уходить. Якщо стан у пацієнта важкий, лікар викликає «швидку» і вона відвозить пацієнта у лікарню. Якщо симптоми у пацієнта не викликають підозри на коронаврус, але були контакти із інфікованими, то пацієнт йде на самоізоляцію. Вивести на екран списки пацієнтів, яким рекомендована самоізоляція і вписаний лікарняний, які направлені на аналіз без лікарняного, яких відвезли і лікарню, яких не обслужили через закінчення часу прийому у лікаря.
5.	5.1 Створити список чисел, що є степенями двійки, задавши їх кількість. Вивести створений список. Виконати такі операції: a) Видалити із списку усі елементи, які знаходяться парних позиціях; b) Інвертувати список, починаючи із заданої позиції; c) Знайти суму елементів на непарних позиціях списку.
	5.2 Написати код, що моделює процес обслуговування покупців в касі магазину. Сформувати список покупців (можна задати їх цифрові ідентифікатори). Відомий час обслуговування касиром кожного покупця. Визначити час перебування кожного покупця у черзі, а також номер покупця, обслуговування якого потребує найменше часу. Вивести на екран сценарій процесу обслуговування (перебування у черзі) покупців.
6.	6.1. Створити список трикутних чисел, задавши їх кількість. Трикутні числа складають послідовність 1, 3, 6, 10, 15, 21, 28, 36, 45, …. Формула для обчислення трикутного числа: $t_1 = 1$; $t_n = n + t_{n-1}$. Вивести створений список. Виконати такі операції: a) Створити підсписок з елементів, кратних 5; b) Видалити із списку усі елементи, що є кратні 5; c) Перевірити, чи дорівнює сума двох послідовних трикутних чисел повному квадрату числа. Підрахувати кількість повних квадратів чисел, що утворює список
	6.2 Написати код, що моделює процес управління списком справ на період (день, тиждень, місяць тощо) – сценарій Dropbox Paper, Google Tasks. Нехай існує перелік завдань, яким задані пріоритети та Deadlines. Потрібно утворити список першочергових справ враховуючи пріоритети та терміни виконання. Задачі можуть бути розбиті на підзадачі, які утворюють підсписки. Відсортувати список за термінами виконання та за пріоритетами. Продемонструвати сценарій виконання завдань, вибираючи їх зі списку, вилучаючи зі списку завдання, які вже виконані, пересуваючи завдання у списку відповідно до зміни пріоритету або терміну виконання.
7.	7.1. Створити список квадратних чисел, задавши їх кількість: 1, 4, 9, 16, 25, 36, 49,…. Вивести створений список. Виконати такі операції: a) Створити підсписок з парних елементів списку квадратних чисел; b) Підрахувати кількість елементів, що є кратні 4; c) Перевірити, що квадратні числа є сумою двох послідовних трикутних чисел 1, 3, 6, 10, 15, 21, 28, 36, 45. Формула для обчислення трикутного числа: $t_1 = 1$; $t_n = n + t_{n-1}$.
	7.2 Написати код, що моделює процес обслуговування клієнтів call-центра. Дзвінки клієнтів формують список (чергу), в якій вони мають порядкові номери. В залежності від кількості операторів список вхідних дзвінків розбивається на декілька списків. Дзвінок направляється до того оператора, список якого найкоротший на момент поточного дзвінка. По мірі обслуговування дзвінків вони видаляються із вхідного списку та додаються до списків операторів, які їх обслуговуватимуть. Вивести на екран сценарій обслуговування операторами дзвінків клієнтів.
8.	8.1 Створити список чисел, які є числами Фібоначчі, задавши їх кількість. Числа Фібоначчі складають послідовність 1, 1, 2, 3, 5, 8, …, в якій кожне наступне число дорівнює сумі двох попередніх. Вивести створений список. Виконати такі операції: a) Визначити елемент списку, вказавши його позицію в списку; b) Довести властивість: $f_1 + f_2 + ... + f_n = f_{n+2} - 1$, де f_n – позначає *n*-ий член послідовності Фібоначчі;

	c) Сформувати новий список з елементів попереднього, які є простими числами
	8.2. Написати код, що моделює процес управління проектом за гнучкою методологією Scrum. Беклог продукту складає список задач з їх пріоритетами. Пріоритети складають ряд Фібоначчі. Планування спринту передбачає створення нового списку задач із беклогу, які будуть виконуватися протягом спринту заданої тривалості. Завдання, які вибрані для спринту, видаляються із беклогу. Після виконання спринту підсумовуються бали тих задач, які були реалізовані. Планується новий спринт. Вивести на екран сценарій управління проектом за методологією Scrum.
9.	9.1. Створити список кубів натуральних чисел (натуральні числа >=1), задавши їх кількість. Вивести створений список. Виконати такі операції: a) Визначити елементи списку, які є двозначними та тризначними числами; b) Видалити двозначні та тризначні числа зі списку; c) Знайти суму чисел в списку, що залишилися.
	9.2. Написати код, що моделює систему масового обслуговування (СМО). У випадкові моменти часу надходять заявки (запити, вимоги) на обслуговування, при цьому заявки, що надійшли, обслуговуються за допомогою наявних у розпорядженні системи каналів обслуговування. Надійшовши в систему обслуговування заявка приєднується до черги вимог, які раніше надійшли. Канал обслуговування вибирає заявку з тих, що знаходяться в черзі, для того, щоб приступити до її обслуговування. Після завершення процедури обслуговування чергової вимоги канал обслуговування приступає до обслуговування наступної заявки, якщо така є в черзі. Цикл функціонування системи масового обслуговування повторюється. При цьому передбачається, що перехід системи на обслуговування чергової вимоги після завершення обслуговування попередньої вимоги відбувається миттєво, у випадкові моменти часу. Вивести на екран сценарій роботи СМО.
10.	10.1. Створити список випадкових цілих чисел, задавши їх кількість. Вивести створений список. Виконати такі операції: a) Визначити елементи списку, які є простими числами; b) Створити новий список, в якому на початку списку розташовані прості числа, потім числа складені. c) Замінити кожне третє число в списку на значення 0 (нуль)
	10.2 Написати код, що моделює роботу автозаправної станції. На АЗС два однакові стовпчики, продуктивність кожної з яких відома. Перший стовпчик обслуговує 1 машину в годину, другий — 3 машини в годину. Є місце, де машини можуть очікувати обслуговування. Якщо стовпчики зайняті, то на цьому місці можуть очікувати обслуговування інші машини, але не більше двох одночасно. Черга обслуговування загальна. Як тільки один зі стовпчиків звільниться, то перша машина із черги може зайняти її місце на колонку (при цьому друга машина просувається на перше місце в черзі). Якщо з'являється третя машина, а всі місця (їх два) у черзі зайняті, то їй відмовляють в обслуговуванні, тому що стояти на дорозі забороненої. Вивести на екран сценарій роботи автозаправної станції.
11.	11.1. Створити список досконалих натуральних чисел, задавши максимальне значення числа (10000). Досконале число – це натуральне число, яке дорівнює сумі всіх своїх дільників. Вивести створений список. Виконати такі операції: a) Визначити елементи списку, які є факторіалами чисел, та підрахувати їх кількість; b) Видалити елементи списку, які є факторіалом числа; c) Знайти елементи, остання цифра яких дорівнює n (задається користувачем).
	11.2 Написати код, що моделює процес обслуговування пасажирів маршрутками. Пасажири утворюють список, їх ідентифікація за порядковими номерами. У маршрутку можуть увійти k людей. На маршруті m машин. Машини прибувають на зупинки кожні n хвилин. Черга пасажирів зменшується на k людей, коли прибуває маршрутка, але протягом чекання наступної машини збільшується на p<=k людей. Вивести на екран сценарій обслуговування пасажирів маршрутками. Визначити, скільки людей перевезуть m маршруток за r циклів роботи.

12.	12.1. Створити список пірамідальних чисел, задавши їх кількість. Формула для *n*-го пірамідального числа: $P_n = \frac{n \times (n+1) \times (n+2)}{6}$ (1, 4, 10, 35,…). Вивести створений список. Виконати такі операції: a) Перевірити, що *n*-е пірамідальне число дорівнює сумі *n* трикутних чисел. Трикутне число визначається за формулою $t_1 = 1$, $t_n = n + t_{n-1}$, $n = \overline{1,...,\infty}$, (1, 3, 6, 10, 15,…); b) Сформувати підсписок з чисел списку, які кратні 5; c) Видалити зі списку усі непарні числа.
	12.2. Написати код, що моделює процес роботи конвеєра. На комплектуючий конвеєр збирального цеху в середньому через 10 хвилин поступають 10 деталей 1-го типу і в середньому через 40 хвилин поступають 40 деталей 2-го типу. Конвеєр складається з секцій, які вміщають по 20 деталей кожного типу. Комплектація починається тільки при наявності деталей обох типів у потрібній кількості і продовжується 20 хвилин. При нестачі деталей секція конвеєру залишається пустою. Визначити ймовірність пропуску секції, середньої довжини черг по кожному типу деталей. Вивести на екран сценарій роботи конвеєра.
13.	13.1 Створити список п'ятикутних чисел, задавши їх кількість. П'ятикутні числа складають послідовність 1, 5, 12, 22, 35, 51,…. Формула для обчислення *n*-го п'ятикутного числа $P_n = 1/2 \times n \times (3 \times n - 1)$. Вивести створений список. Виконати такі операції: a) Замінити значення елементів на парних позиціях на їх подвоєне значення; b) Знайти середнє арифметичне чисел списку c) Поділити список на два рівних по кількості елементів. Значення останнього елемента першого списку та першого елемента другого списку зробити однаковими.
	13.2 Написати код, що моделює процес роботи вантажного аеропорту. Вантажі прибувають для відправлення в аеропорт у контейнерах із інтенсивністю два контейнери за 1 хвилину. Вантажний аеропорт не має фіксованого розкладу, а літаки відправляються по мірі їх повного завантаження. У розпорядженні є два типи літаків для перевезення вантажів: три літаки з вантажопідіймальністю 80 контейнерів і два літаки з вантажопідіймальністю 140 контейнерів. Час польоту кожного літака в середньому 3 години. Керуючий аеропортом намагається якнайчастіше використовувати літаки меншої вантажопідіймальності. Літаки, що піднімають 140 контейнерів, використовуються тільки тоді, коли інших немає в наявності. Припускається, що часом вантаження можна не враховувати. Визначити середній час очікування контейнерів із вантажами та середнє завантаження літаків обох типів. Вивести на екран сценарій роботи аеропорту.
14.	14.1. Створити список шестикутних чисел, задавши їх кількість. Шестикутні числа складають послідовність 1, 6, 15, 28, 45,…. Формула для обчислення *n*-го шестикутного числа $P_n = 2 \times n^2 - n$. Вивести створений список. Виконати такі операції: a) Знайти суму парних елементів списку; b) Замінити в списку значення, кратні 5, на значення, помножені на 10. c) Сформувати підсписок із чисел, кратних 10.
	14.2. Написати код, що моделює процес роботи роздрібної торгівлі. Фірма має в місті 6 точок роздрібного продажу. Попит на товари у цих точках 10 од. товару в день. Торгові точки обслуговуються оптовим магазином. На передачу запиту торгової точки в магазин потрібно 1 день. Товари за запитом поступають з оптового магазина в торгову точку в середньому через 5 днів після одержання запиту. Оптовий магазин кожні 14 днів розміщує замовлення на фабриці. Час, протягом якого магазин одержує вантаж із фабрики, в середньому складає 90 днів. Визначити обсяг запасу товару в оптовому магазині, ймовірність невдоволеного запиту торгової точки. Вивести на екран сценарій роботи роздрібної торгівлі.
15.	15.1. Задати дату народження у вигляді ДД.ММ.РРРР. Створити список, елементи якого складають квадрат Піфагора, що побудований за датою народження. Алгоритм побудови квадрата Піфагора: http://www.xsp.ru/online/pifagor/description.php. Значення, які в квадраті Піфагора відсутні, в списку записати нулями. Вивести створений список. Виконати такі операції: a) Підрахувати кількість нульових елементів; b) Знайти елементи, які містять по *n* (1,2,3,або 4 - задане користувачем) цифр; c) Сформувати новий список з елементів з найменшою та з найбільшою кількістю цифр.

	15.2. Написати код, що моделює процес роботи банку. Банк має 5 кас для обслуговує клієнтів протягом робочого дня (5 годин). Інтервали часу між надходженням клієнтів складають 1 хвилину. Час обслуговування клієнтів 5 хвилин. До кожної каси формується окрема черга. Клієнт, що надійшов на обслуговування, обирає найкоротшу чергу, при цьому, якщо найкоротших черг декілька, то клієнт обирає першу. В зв'язку з тим, що керівництво банку цікавлять поточні витрати та якість послуг, що надаються клієнтам, ставиться питання про можливість зменшення кількості кас. Вивести на екран сценарій роботи кас банку та кількість кас, які залишаться в банку.
16.	16.1. Створити список з додатних та від'ємних чисел, задавши їх кількість. Вивести створений список. Виконати такі операції: a) Замінити від'ємні числами їх квадратами; b) Упорядкувати список за спаданням; c) Визначити суму парних чисел.
	16.2. Написати код, що моделює процес роботи ліфта в п'ятиповерховому офісі. Люди надходять на перший поверх кожні 2 хвилини. Для кожного з інших поверхів ймовірність того, що людина прямує саме на цей поверх, дорівнює 0,25. Час, потрібний для переміщення ліфту на один поверх складає 15 секунд. Час, потрібний для завантаження та розвантаження ліфту, не враховується. На певному поверсі людина залишається протягом години. Коли людина залишає поверх (2,3,4,5), то з ймовірністю 0,7 вона прямує на перший поверх і з ймовірністю 0,1 прямує на один із інших поверхів. Місткість ліфту – 6 людей. Якщо людина не вміщується у ліфт, то вона залишається в очікуванні ліфту на своєму поверсі. На початку моделювання ліфт знаходиться на першому поверсі. Визначити: середній час очікування в усіх чергах (на кожному поверсі у двох напрямках вверх і униз); максимальну та середню кількість людей у ліфті. Вивести на екран сценарій роботи ліфта.
17.	17.1. Створити два списки цілих чисел. Вивести їх на екран. Виконати такі операції: a) Побудувати третій список, в якому кожний елемент дорівнює найбільшому спільному дільнику відповідних елементів введених списків. b) Визначити елементи, які співпадають в обох списках та надрукувати їх; c) Видалити з першого списку елементи, які співпадають з елементами другого списку;
	17.2. Написати код, що моделює процес зарахування абітурієнтів на магістерську освітню програму. Формується список з k абітурієнтів, що подали заяви і призначені на іспит. Після написання іспиту формується новий список, елементами якого є пари (номер абітурієнта . екзаменаційна оцінка). Другий список сортується за екзаменаційною оцінкою. Виділено m бюджетних місць ($m < k$). З другого списку формується третій список, в якій переносяться перші m елементів другого списку. Ці елементи видаляються з другого списку. В процесі формування третього списку n абітурієнтів з прохідними балами забирають документи, тобто n елементів видаляються з третього списку. Для доповнення третього списку до кількості в m елементів з другого списку n елементів з позиції m+1 переносяться у список 3. Вивести на екран сценарій зарахування абітурієнтів.
18.	18.1 Створити список чисел або символів. Виконати такі операції: a) Обчислити кількість елементів у найдовшій серії, вивести усі серії та шукану кількість. Серія — це послідовність однакових елементів, розташованих поспіль; b) В кожній серії списку залишити по одному елементу та надрукувати список ; c) Додати на початок списку n елементів, кратних заданому користувачем числу.
	18.2. Написати код, що моделює процес прийняття фахівця на роботу. На вакантну посаду подано k резюме, з яких сформовано список. Попередній аналіз резюме визначив пріоритети резюме. В результаті сформований новий список, в якому резюме відсортовані за пріоритетом. Вищий пріоритет має нижчий номер. На співбесіду допущено m (m<k) резюме. В процесі проходження 1 туру співбесіди n (n<m) людей отримали відмову. В процесі проходження 2 туру відмовлено p <m-n людям. Решту людей взяли на випробувальний термін. Вивести на екран сценарій модифікації списків в процесі походження 1,2 турів та випробувального періоду. Запропонувати сценарій вибору одної людини після випробувального періоду.

19.	19.1. Створити числовий список, кількість елементів якого задана користувачем. Значення елементів в списку можуть повторюватися. Вивести список на екран. Виконати такі операції: a) Визначити позицію першого та останнього входження заданого числа в список. b) Знайти елемент, значення якого повторюється найбільшу кількість разів. c) Видалити елементи, що повторюються, залишивши в списку значення, що входять по одному разу.
	19.2. Написати код, що моделює процес навчання студентів ЗВО з певної дисципліни (за основу взяти свій власний досвід). Студент отримує перелік завдань для семестрового зарахування з певної дисципліни. Сформувати список з пар (ID_завдання . кількість балів, які можна отримати). На початку моделювання сумарна кількість балів дорівнює 0. В процесі навчання студент здає завдання, накопичує семестрові бали. Формується новий список пар (ID-завдання . кількість балів, які студент отримав), модифікується перший список, з якого вилучаються виконані та зараховані завдання. Вивести на екран сценарій модифікації списків в процесі навчання в семестрі.
20.	20.1. Створити цілочисловий список з додатних, від'ємних та нульових елементів, кількість елементів списку задана користувачем. Вивести список на екран. Виконати такі операції: a) Сформувати підсписок з елементів між першим від'ємним та нульовим елементом та між першим додатним та нульовим; b) Видалити нульові елементи списку; c) Здійснити пошук елемента, значення якого є факторіалом числа
	20.2. Написати код, що моделює процес вантаження та розвантаження кораблів в порту. Кораблі надходять до гавані кожні 2 дні. У гавані є док з двома якірними стоянками та двома кранами для розвантаження кораблів. Кораблі, що надійшли у момент, коли обидві якірні стоянки зайняті, очікують у черзі з дисципліною обслуговування FIFO, Час, потрібний для розвантаження одним краном одного корабля, 1 дня. Якщо у гавані тільки один корабель, розвантаженням його займаються обидва крани і час розвантаження (час, що залишився) зменшується вдвічі. Якщо надходить ще один корабель, то один кран без затримки відправляється його розвантажувати. Інший кран продовжує розвантажувати корабель, що надійшов раніше, а час, що залишився на його обслуговування збільшується вдвічі. Створити списки, які містять дані про надходження кораблів та їх обслуговування. Визначити мінімальний, максимальний та середній часу обслуговування кораблів у гавані. Вивести на екран сценарій модифікації списків.

Лабораторна робота 5
Обробка раціональних та комплексних чисел мовами функціонального програмування

Мета

Опанувати технологію абстракції даних в мовах функціонального програмування. Реалізувати програму обробки раціональних та комплексних чисел мовами функціонального програмування, представивши ці числа конструкціями типу «пара»

Рейтингова таблиця лабораторної роботи 5

Вид роботи	**Кількість балів**	Deadline
Код задачі 1	3	листопад
Код задачі 1	3	
Звіт	1	
Якість		
Разом	**7**	

Теоретичні відомості

Раціональне рівняння - це рівняння виду f(x) = g(x), де f(x) і g(x) – раціональні вирази. Раціональні вирази – це цілі і дробові вирази, з'єднані між собою знаками арифметичних дій: ділення, множення, додавання або віднімання, зведення в цілу степінь і знаками послідовності цих виразів.

Якщо хоча б в одній частині раціонального рівняння міститься дріб, то рівняння називається дрібно-раціональним.

Алгоритм розв’язання цілих раціональних рівнянь

1. Визначити найменший спільний знаменник для всього рівняння.
2. Визначити множники, на які потрібно помножити кожен член рівняння.
3. Привести до спільного знаменника всі рівняння.
4. Здійснення пошуку коренів отриманого цілого раціонального рівняння.

Алгоритм розв’язання дрібно-раціональних рівнянь

1. Знайти значення змінної, при яких рівняння не має сенс (ОДЗ);
2. Перетворити рівняння до виду алгебраїчного дробу f(x)/g(x):
   2.1.Знайти спільний знаменник дробив, що входять в рівняння;
   2.2.Помножити обидві частини рівняння на спільний знаменник;
   2.3.Перенести всі члени рівняння в один бік.
3. Прирівняти чисельник до нуля і розв’язати рівняння f(x) = 0.
4. Прирівняти знаменник до нуля і розв’язати рівняння g(x) = 0
5. Якщо корені чисельника і знаменника співпали, то їх слід виключити з результату. Відповідь містить корені чисельника, які не обертають в нуль знаменник.

Арифметика раціональних чисел

$$\frac{n_1}{d_1} + \frac{n_2}{d_2} = \frac{n_1 d_2 + n_2 d_1}{d_1 d_2}$$

$$\frac{n_1}{d_1} - \frac{n_2}{d_2} = \frac{n_1 d_2 - n_2 d_1}{d_1 d_2}$$

$$\frac{n_1}{d_1} \cdot \frac{n_2}{d_2} = \frac{n_1 n_2}{d_1 d_2}$$

$$\frac{n_1/d_1}{n_2/d_2} = \frac{n_1 d_2}{d_1 n_2}$$

$\frac{n_1}{d_1} = \frac{n_2}{d_2}$ тогда и только тогда, когда $n_1 d_2 = n_2 d_1$

Арифметика комплексних чисел

$$r = |z| = \sqrt{a^2 + b^2}$$

$$z = r(\cos\phi + i\sin(\phi))$$

$$\cos\varphi = \frac{a}{r}$$

$$\sin\varphi = \frac{b}{r}$$

Теоретичні відомості по комплексним числам див. https://100task.ru/sample/137.aspx та в презентації лек7 арифметика комплексних чисел в Scheme.pptx

Приклади коду **(Racket)**

Арифметика раціональних чисел

```
(define (numer x) (car x))                ;чисельник
(define (denom x) (cdr x))                ;знаменник
(define (make-rat n d) (cons n d))        ; створення пари
(define(print-rat x)                      ; друк пари
  (newline)
  (display (numer x))
  (display "/")
  (display (denom x)))

(make-rat -5 6)

(print-rat (make-rat -5 6))          ;виклик друку пари у вигляді дробу
;================================
;операції з дробами
;================================
(define fraction1 (make-rat 4 7) )          ;створення дробу 1
(define fraction2 (make-rat 5 9) )          ;створення дробу 2
;===========додавання дробив==============
(define (add-rat x y)
    (make-rat (+ (* (numer x) (denom y)
                 )
                 (* (numer y) (denom x)
                 )
              )
              (* (denom x) (denom y)
```

```
            )
      )
   )
 (print-rat (add-rat fraction1 fraction2))
;==========віднімання дробив==============
(define (sub-rat x y)
      (make-rat (- (* (numer x) (denom y))
                   (* (numer y) (denom x)))
                (* (denom x) (denom y))))

(print-rat (sub-rat fraction1 fraction2))
;==============ділення дробив======================
(define (div-rat x y)
       (make-rat (* (numer x) (denom y))
                 (* (denom x) (numer y))))

(print-rat (div-rat fraction1 fraction2))
;==============множення дробив======================
(define (mul-rat x y)
        (make-rat (* (numer x) (numer y))
                  (* (denom x) (denom y))))
(print-rat (mul-rat fraction1 fraction2))
;==============порівняння дробив======================
(define (equal-rat? x y)
        (= (* (numer x) (denom y))
           (* (numer y) (denom x))))
(newline)
(equal-rat? fraction1 fraction2)
;==============створення та скорочення дробу============
(define (make-rat n d)
   (let ((nod (gcd n d)))
       (cons (/ n nod) (/ d nod))))
```

Арифметика комплексних чисел

```
;==========селектори дійсної та уявної частин компл числа=====
(define (Myreal-part z) (car z))
(define (Myimag-part z) (cdr z))
;========== тригонометрична форма компл числа======== ==
(define (square x)
            (* x x))
(define (magnitude1 z)
            (sqrt (+ (square (Myreal-part z)) (square (Myimag-part z)))))
(define (angle1 z)
            (atan (Myimag-part z) (Myreal-part z)))
(define (make-from-real-imag x y)
            (cons x y))
(define (make-from-mag-ang r a)
            (cons (* r (cos a)) (* r (sin a))))

(make-from-real-imag 2 3)
;======= додавання та віднімання компл чисел в алгебраїчній формі======
(define (add-complex z1 z2)
 (make-from-real-imag (+ (Myreal-part z1) (Myreal-part z2))
   (+ (Myimag-part z1) (Myimag-part z2))))

(define (sub-complex z1 z2)
   (make-from-real-imag (- (Myreal-part z1) (Myreal-part z2))
     (- (Myimag-part z1) (Myimag-part z2))))
```

```
(define complex1 (make-from-real-imag 2 3))     ;створення комплексного числа1
(define complex2 (make-from-real-imag -5 1))

(add-complex complex1 complex2)
(sub-complex complex1 complex2)
;=====множення та ділення компл чисел в тригонометричній формі======
(define (mul-complex z1 z2)
      (make-from-mag-ang (* (magnitude1 z1) (magnitude1 z2))
                  (+ (angle1 z1) (angle1 z2))))

(define (div-complex z1 z2)
      (make-from-mag-ang (/ (magnitude1 z1) (magnitude1 z2))
                  (- (angle1 z1) (angle1 z2))))
 (mul-complex complex1 complex2)
 (div-complex complex1 complex2)
```

Зміст звіту

1. Титульний лист
2. Мета роботи
3. Умова завдання
4. Аналіз задачі та математичні забезпечення для розв'язання (у випадку математичної задачі)
5. Обґрунтування вибору мови функціонального програмування та IDE
6. Код
7. Скріншот роботи та результату
8. Оцінка достовірності результату (перевірка правильності результату)
9. Висновок

Завдання до лабораторної роботи 5

1. **Написати процедури, що обробляють раціональні числа (працюють з дробами), які подати у вигляді чисельника і знаменника .**
2. **Написати процедури, що обробляють комплексні числа, які слід подати в декартовому (алгебраїчному) (z = a + i×b) та/або полярному (z=√(a^2+b^2)(cosϕ + i×sin(ϕ))) зображенні.**

Номер варіанта	Умова завдань
1.	1.1 Задати декілька (два) раціональних (дробових) чисел і знайти їх найменше спільне кратне (НСК). Для цього знайти НСК чисельників цих дробив і поділити на найбільший спільний дільник (НСД) знаменників цих дробив. НСК(a/b,c/d)=НСК(ad/bd,bc/bd)=НСК(НСК(ad,bc)/bd)=НСК(ad,bc)/НСД(bd,НСК(ad,bc))
	1.2. Створити список з комплексних чисел, заданих в алгебраїчній формі $z = a + ib$. Створити новий список, елементи якого є комплексні числа, переведені з алгебраїчної форми у тригонометричну. $z = r(\cos\phi + i\sin(\phi)), r = \mid z \mid = \sqrt{a^2 + b^2}, \cos\varphi = \frac{a}{r}, \sin\varphi = \frac{b}{r}$ Надрукувати новий список.
2. і	2.1 Знайти суму чотирьох дробив $\frac{1}{a}+\frac{1}{b}+\frac{1}{c}+\frac{1}{d}$. де *a, b, c, d* – непарні натуральні числа. Якщо сума цих дробив дорівнює 1, вивести повідомлення "YES", інакше "NO". Для розв'язання задачі написати процедуру приведення дробив до спільного знаменника.

	2.2 Створити список з комплексних чисел, заданих в алгебраїчній формі в декартових координатах ***z=a + ib.*** Створити новий список, елементи якого є комплексні числа, переведені з декартових координат у полярні. Комплексне число в полярних координатах складається з модуля $\mid z \mid = \sqrt{a^2 + b^2}$ та аргументу $\arg z = arctg\frac{b}{a}$. Надрукувати новий список.
3.	3.1. Написати програму, яка моделює та виводить рівняння та його числовий розв'язок для задачі: «Летіла зграя гусей, а назустріч їм один гусак і каже: «Здрастуйте, сто гусей!» - «Нас не сто гусей, - відповідає йому ватажок стада, - якби нас було стільки, скільки тепер, та ще стільки, та півстільки, та чверть стільки, та ще ти, гусак, з нами, тоді б нас було сто гусей». Скільки гусей було в зграї?»
	3.2 Створити список з комплексних чисел, які задані в алгебраїчній формі ***a + ib.*** Надрукувати список. Обчислити суму комплексних елементів списку
4.	4.1 Написати програму, яка демонструє опрацювання операцій з дробами в процесі розв'язання такої задачі. «Троє друзів виграли деяку суму грошей. Частка першого склала x цієї суми, частка другого склала y від суми, третьому досталося z грошей. Яким був виграш?» Значення x та y задавати у вигляді дробив. Значення z – ціле число. Контрольний приклад: x=1/4 від суми грошей, y=1/7 від суми грошей, z = 17 грошей.
	4.2 Створити два списки цілих чисел, кількість елементів в яких однакова. Створити третій список, елементами якого є комплексні числа в алгебраїчній формі ***a + ib,*** дійсна частина яких є числами з першого списку, уявна частина – це числа з другого списку. Надрукувати створений список комплексних чисел.
5.	5.1 Написати програму, яка демонструє опрацювання операцій з дробами в процесі розв'язання такої задачі. «Подорожній запитав «Чи далеко до села, що попереду?» Отримав відповідь: «Відстань від твого села дорівнює 1/3 всієї відстані між селами. А якщо пройдеш ще дві версти, будеш на ½ (посередині) між селами». Скільки верст залишилося йти подорожньому?»
	5.2. Створити список з комплексних чисел, які задані в алгебраїчній формі ***a + ib.*** Надрукувати список. Обчислити добуток комплексних елементів списку.
6.	6.1. Створити список, елементами якого є раціональні числа у вигляді дробив. Упорядкувати список за зростанням (спаданням). Для розв'язання задачі привести усі дроби до спільного знаменника і порівнювати чисельники. Надрукувати список після приведення елементів до спільного знаменника. Наприклад, задані 7/20, 1/5, 3/10 ⇒7/20, 4/20, 6/20 ⇒ 4/20, 6/20, 7/20.
	6.2 Створити список з трьох комплексних чисел, які задані в алгебраїчній формі ***a + ib.*** Розв'язати комплексне рівняння, коефіцієнтами якого при x та y та вільний член є комплексні числа зі списку. Контрольний приклад: $(1 + 2i)x+(3 - 5i)y = 1 - 3i$. Відповідь $x = -\frac{4}{11}; y = \frac{5}{11}$
7.	7.1. Створити список, елементами якого є раціональні числа у вигляді дробив. Обчислити суму та добуток елементів списку. Для обчислення суми привести усі дроби до спільного знаменника. Надрукувати список після приведення елементів до спільного знаменника. Наприклад, створений список дробив: 7/20, 1/5, 3/10 ⇒ 7/20, 4/20, 6/20 ⇒ (7+4+6)/20 ⇒ 17/20.
	7.2 Створити список, елементами якого є комплексні числа в алгебраїчній формі ***a + ib.*** Побудувати новий список, елементами якого є квадрати комплексних чисел з першого списку. Надрукувати отриманий список.
8.	8.1 Написати програму, яка демонструє опрацювання операцій з дробами в процесі розв'язання такої задачі. «Чоловік випив 1/6 чашечки чорної кави і долив її молоком. Потім він випив 1/3 чашечки і знову долив її молоком. Потім він випив 1/2 чашечки і знову долив її молоком. Нарешті, він випив повну чашку. Чого він випив більше – чорної кави чи молока?»

	8.2. Визначити геометричне місце точок (г.м.т.), що зображують комплексне число в декартовій системі координат за заданим модулем $\|z\| = \sqrt{a^2 + b^2}$ та аргументом $\arg z = arctg\frac{b}{a}$. Відповідно до геометричного подання комплексних чисел геометричним місцем точок можуть бути частини кола, радіус і центр яких слід визначити та надрукувати. Контрольний приклад: при модулі комплексного числа |z| = 1 г.м.т. – коло радіуса 1 з центром (0,0).
9.	9.1. Створити список з парною кількістю елементів, які є різними раціональними числами у вигляді дробив. Для кожної пари елементів списку вивести повідомлення про їх рівність або нерівність. Для розв'язання задачі привести усі дроби до спільного знаменника. Надрукувати список після приведення елементів до спільного знаменника.
	9.2. Створити список з парною кількістю елементів, які є комплексними числами в алгебраїчній формі ***a + ib***. Створити новий список, елементами яких є добутки кожної пари комплексних чисел першого списку. Надрукувати новий список.
10.	10.1. Написати програму, яка моделює роботу вірусної програми за таким сценарієм.. Вірус знищує дані з пам'яті комп'ютера. За першу секунду вірус знищив 1/2 даних, за другу секунду – 1/3 залишку, за третю секунду – 1/4 того, що зберіглося, за четверту секунду – 1/5 залишку. На п'ятій секунді його дію припинила антивірусна програма. Яка частина даних вціліла? Відповідь подати у вигляді дробу. Контрольна відповідь 1/5.
	10.2 Створити список комплексних чисел, заданих в алгебраїчній формі ***a + ib***. Переписати в новий список ті комплексні числа з першого списку, які в геометричній інтерпретації зображують точки в четвертому квадранті декартової системи координат.
11.	11.1.Створити список. елементами якого є різні раціональні числа у вигляді дробив. Обчислити кількість елементів в списку, які є нескоротними дробами та надрукувати їх. Для перевірки дробу на скоротність обчислити найбільший спільний дільник (НСД). Якщо НСД дорівнює одиниці, то дріб є нескоротний. Якщо НСД відмінний від одиниці, то дріб скоротний.
	11.2 Створити список комплексних чисел, заданих в тригонометричній формі . Переписати в новий список ті комплексні числа з першого списку, які в геометричні інтерпретації зображують точки в третьому квадранті декартової системи координат.
12.	12.1. Створити список з парною кількістю елементів, які є різними раціональними числами у вигляді дробив. Для кожної пари елементів списку визначити їх різницю і записати в новий список. Надрукувати список, утворений з різниць елементів. Для розв'язання задачі привести усі дроби до спільного знаменника.
	12.2. Створити список з парною кількістю елементів, які є комплексними числами в алгебраїчній формі ***a + ib***. Створити новий список, елементами якого є суми кожної пари комплексних чисел першого списку. Надрукувати новий список.
13.	13.1 Створити список з парною кількістю елементів, які є різними раціональними числами у вигляді дробив. Для кожної пари елементів списку виконати ділення дробив, результат якого записати у новий список. Надрукувати список, утворений з результатів ділення кожної пари. Для розв'язання задачі необхідно перевернути дріб, на який ділимо, після чого здійснити множення дробив.
	13.2 Створити список з парною кількістю елементів, які є комплексними числами в алгебраїчній формі ***a + ib***. Створити новий список, елементами якого є добутки кожної пари комплексних чисел першого списку. Надрукувати новий список.
14.	14.1. Створити список з парною кількістю елементів, які є різними раціональними числами у вигляді дробив. Для кожної пари елементів списку виконати множення дробив, результат якого записати у новий список. Надрукувати список, утворений з результатів множення кожної пари.

	14.2. Створити список з парною кількістю елементів, які є комплексними числами в алгебраїчній формі ***a + ib***. Створити новий список, елементами якого є частки від ділення кожної пари комплексних чисел першого списку. Надрукувати новий список.
15.	15.1. Розв'язати ціле раціональне рівняння $\frac{x-1}{2}+\frac{2x}{3}=\frac{5x}{6}$, виконавши операції визначення найменшого спільного знаменника дробив, приведення рівняння до спільного знаменника, пошуку кореня в перетвореному рівнянні.
	15.2. Створити список цілих додатних і від'ємних чисел, з кожної пари яких побудувати комплексні числа у тригонометричній формі і записати їх до нового списку. Тригонометрична форма комплексного числа задається модулем *z* і аргументом $\arg z = arctg\frac{b}{a};\ z = r(\cos\phi + i\sin(\phi)), r = \mid z \mid = \sqrt{a^2+b^2}, \cos\varphi = \frac{a}{r}, \sin\varphi = \frac{b}{r}$
16.	16.1. Розв'язати дрібно-раціональне рівняння $\frac{5x-3}{x-3}=\frac{2x-3}{x}$, виконавши операції приведення дробив до спільного знаменника, прирівнювання чисельника до нуля і пошук його коренів, прирівнювання знаменника до нуля і пошук його коренів, порівняння коренів чисельника і знаменника, вилучення тих, що співпадають. (Контрольна відповідь $x = -1$).
	16.2. Задати два комплексних числа z_1, z_2 . Виконати операції додавання, віднімання, множення та ділення в алгебраїчній та тригонометричній формах подання цих чисел. Алгебраїчна форма: ***a + ib***. Тригонометрична форма: $z = r(\cos\phi + i\sin(\phi)), r = \mid z \mid = \sqrt{a^2+b^2}, \cos\varphi = \frac{a}{r}, \sin\varphi = \frac{b}{r}$
17.	17.1. Створити список елементи якого є дійсні десяткові дроби типу 2.5. Утворити новий список, елементами якого є раціональні числа у вигляді правильних (< 1) і неправильних (>1) дробив, отриманих в результаті переведення десяткових дробив у раціональні дроби. Наприклад; $2.5 \Rightarrow 2\frac{5}{10} \Rightarrow \frac{25}{10} \Rightarrow \frac{5}{2}$. Надрукувати список, утворений з результатів переведення дробив з десяткової форми у звичайну.
	17.2. Створити список з парною кількістю елементів, які є комплексними числами в алгебраїчній формі ***a + ib***. Створити новий список, елементами якого є відстані між двома комплексними числами з кожної пари першого списку. Відстань між двома комплексними числами визначається як $l = \sqrt{(a_1-a_2)^2+(b_1-b_2)^2}$, де a_1, a_2 – дійсні частини комплексних чисел, b_1, b_2 – уявні частини комплексних чисел.
18.	18.1 Розв'язати дрібно-раціональне рівняння $\frac{3x-5}{-2}=\frac{1}{x}$, виконавши операції приведення дробив до спільного знаменника, прирівнювання чисельника до нуля і пошук його коренів, прирівнювання знаменника до нуля і пошук його коренів, порівняння коренів чисельника і знаменника, вилучення тих, що співпадають. (Контрольна відповідь $x_1 = 1,\ x_2 = \frac{2}{3}$)
	18.2. Створити список комплексних чисел в тригонометричній формі. Перше комплексне число в списку задається: $z = r(\cos\phi + i\sin(\phi)), r = \mid z \mid = \sqrt{a^2+b^2}, \cos\varphi = \frac{a}{r}, \sin\varphi = \frac{b}{r}$. Кожне *i*-те комплексне число в списку обчислюється як *i*-та степінь першого числа. Для зведення комплексного числа в степінь використовувати формулу Муавра: $(\cos\phi + i\sin\phi)^n = (\cos n\phi + i\sin n\phi), \forall n \in N$
19.	19.1. Створити список, елементами якого є раціональні числа у вигляді дробив. Визначити кількість елементів списку, знаменники яких є парними числами. Надрукувати список, утворений з елементів. знаменники яких є парними числами.

	19.2. Створити список з комплексних чисел, заданих в алгебраїчній формі $z = a + ib$. Створити новий список, елементи якого є комплексні числа, переведені з алгебраїчної форми у показову форму комплексного числа: $z = re^{i\phi},\ e^{i\phi} = \cos\phi + i\sin\phi,\ r = \mid z \mid = \sqrt{a^2 + b^2},\ \cos\phi = \frac{a}{r}, \sin\phi = \frac{b}{r}$ Визначити добуток першого та останнього числа списку в показовій формі.
20.	20.1. Створити список, елементами якого є раціональні числа у вигляді дробив. Визначити кількість елементів списку, чисельники яких є парними числами. Надрукувати список, утворений з елементів. чисельники яких є парними числами.
	20.2. Створити список з комплексних чисел, заданих в алгебраїчній формі $z = a + ib$. Створити новий список, елементи якого є комплексні числа, переведені з алгебраїчної форми у показову форму комплексного числа: $z = re^{i\phi},\ e^{i\phi} = \cos\phi + i\sin\phi,\ r = \mid z \mid = \sqrt{a^2 + b^2},\ \cos\phi = \frac{a}{r}, \sin\phi = \frac{b}{r}$ Визначити суму усіх елементів списку в показовій формі.

Лабораторна робота 6
Обробка структур типу векторів і матриць, стеків та черг мовами функціонального програмування

Мета

Опанувати теоретичні основи обробки структур типу векторів і матриць, стеків та черг мовами функціонального програмування та розробити програми їх реалізації

Рейтингова таблиця лабораторної роботи 6

Вид роботи	Кількість балів	Deadline
Код задачі 1	3	листопад
Код задачі 1	3	
Звіт	1	
Якість		
Разом	**7**	

Теоретичні відомості

Вектори

Вектори – це різнорідні структури, елементи яких індексуються цілими числами. Довжина вектору – це кількість елементів, які він містить. Це число є невід'ємним цілим числом, яке фіксується при створенні вектору.

Індекси вектору – це точні невід'ємні цілі числа, менші за довжину вектора. Перший елемент у векторі індексується нулем, останній елемент індексується на одиницю менше довжини вектору.

В мові Scheme (Racket) вектори записуються із позначенням **# (obj ...).** Наприклад, вектор довжиною 3, що містить нульове значення в індексі 0, список (2 2 2 2) в індексі 1 та рядок "Anna" в індексі 2, можна записати так: #(0 (2 2 2 2) "Anna"). Векторні константи повинні quoted (цитуватися) за допомогою символу **'**, наприклад,

' #(0 (2 2 2 2) "Anna") , що означає # (0 (2 2 2 2) "Anna")

Операції, які застосовують до векторів мовою Scheme (Racket), подані в таблиці 6.1

Таблиця 6.1 Процедури обробки векторів мовою Scheme (Racket

Ім'я процедури	Зміст процедури
(vector? obj)	Повертає #t, якщо obj є вектором, інакше повертає #f.
(make-vector k)	Повертає створений вектор з k елементів.
(make-vector k fill)	Повертає створений вектор з k елементів, кожний елемент якого ініціалізований значенням fill.
(vector obj ...)	Повертає створений вектор, елементи якого містять задані аргументи. Аналогічно list.
(vector-length вектор)	Повертає кількість елементів у векторі як точне ціле число.
(vector-ref вектор k)	Повертає значення k- го елемента вектору. k має бути допустимим (валідним) індексом вектору.
(vector-set! вектор k obj)	Присвоєння значення obj k-му елементу вектору
(vector->list vector)	Повертає створений список об'єктів, що містяться в елементах вектору

(list->vector list)	Повертає новостворений вектор, ініціалізований елементами списку .
(vector-fill! вектор заповнювач)	Кожний елемент вектору набуває значення fill

Черги (стеки) в мовах функціонального програмування

Черга (queue) являє собою послідовність, в яку можна додавати елементи з одного кінця (він називається хвостом (rear)) і вибирати з іншого (він називається головою (front)). Оскільки елементи видаляються завжди в тому ж порядку, в якому вони були додані, чергу називають буфером FIFO (англ. First in, first out – першим увійшов, першим вийшов).

Стек — різновид лінійного списку, який працює за принципом «останнім прийшов — першим пішов» (LIFO). Всі операції (наприклад, видалення елементу) в стеку можна проводити тільки з одним елементом, який знаходиться на верхівці стека та був введений в стек останнім.

Стек може бути організований як вектор або список з додатковим зберіганням ще й вказівника на верхівку стека.

Черга може бути реалізована за допомогою вектору або списку з додатковим зберіганням ще й вказівників на початок і кінець черги.

Операції, що застосовують до структур типу черги, подані в таблиці 6.2.

Таблиця 6.2 Процедури обробки черг мовою Scheme (Racket

Ім’я процедури	Зміст процедури
(make-queue)	конструктор повертає порожню чергу (чергу, в якій немає жодного елемента).
(empty-queue? <очередь>)	перевіряє, чи порожня черга
(front-queue <очередь>)	селектор повертає об'єкт, що знаходиться в голові черги. Якщо черга порожня, він повідомляє про помилку. Черга не модифікується
(insert-queue! <очередь> <елемент>)	мутатор вставляє елемент в хвіст черги і повертає в якості значення змінену чергу
(delete-queue! <очередь>)	мутатор видаляє елемент з голови черги і повертає в якості значення змінену чергу. Якщо перед знищенням елемента черга виявляється порожньою, виводиться повідомлення про помилку.

Оскільки черга є послідовністю елементів, її можна представити як звичайний список Головою черги є **car** цього списку, вставка елемента в чергу зводиться до додавання нового елемента в кінець списку, а знищення елемента з черги складається у взятті **cdr** списку.

Чергу представляють у вигляді списку і тримають додатковий покажчик на його останню пару, щоб уникнути перегляду всього списку для пошуку кінця черги. Черга, таким чином, представляється у вигляді пари покажчиків, **front-p**tr і **rear-ptr**, які позначають, відповідно, першу і останню пару звичайного списку.

Приклади коду обробки векторів

```
;===========сума двох векторів===============
(define add-vectors
  (lambda (vec-1 vec-2)
    (let* ((len (vector-length vec-1))  ; The vectors should
                                        ; have the same length.
           (result (make-vector len)))
      (do ((index 0 (+ index 1)))
          ((= index len) result)
        (vector-set! result index
                     (+ (vector-ref vec-1 index)
                        (vector-ref vec-2 index)))))))

(newline)
```

```
(add-vectors '#(3 5 7 9) '#(3 1 4 1))
;==========сума елементів вектора================
(define vector-sum
  (lambda (vec)
    (let ((len (vector-length vec))
          (result 0))
      (do ((index 0 (+ index 1)))
          ((= index len) result)
        (set! result (+ result (vector-ref vec index)))))))

(newline)
(vector-sum '#(3 5 7 9))
;================об'єднання векторів==================
(define vector-append
  (lambda vecs
    (let* ((len (apply + (map vector-length vecs)))
           (result (make-vector len)))
      (let loop ((result-index 0)
                 (source-index 0)
                 (rest-of-vecs vecs))
        (cond ((null? rest-of-vecs) result)
              ((= source-index (vector-length (car rest-of-vecs)))
               (loop result-index 0 (cdr rest-of-vecs)))
              (else
               (vector-set! result result-index
                            (vector-ref (car rest-of-vecs) source-index))
               (loop (+ result-index 1) (+ source-index 1) rest-of-vecs)))))))

(vector-append '#(3 1 4 1 6) '#(a b c)'#(1 2 3 4))
```

Приклади коду обробки черг

```
;;#######Implimenting Queue#########
(define (make-queue)
 (define p (cons '() '() ) )
 (cons p p)
)
;========== ==Перевірка черги на пустоту==========
(define (null-queue? q)
 (and
  (eq? (front q) (rear q)) (eq? (car (front q)) '() ))
)
;===== селектор (доступ) до першого елемента черги===
(define (front q)
 (car q))
;==== селектор (доступ) до останнього елемента черги ==
(define (rear q)
 (cdr q))
;========додавання нового елемента в чергу========
(define (push q e)
 (define p (cons e '()))
 (if (null-queue? q)
  (begin (set-car! q p)
   (set-cdr! q p)
  )
  (begin
   (set-cdr! (rear q) p)
   (set-cdr! q p)
  ) ) )
;========== вилучення елемента з черги============
(define (pop q)
 (define x 0)
 (if (null-queue? q)
  'Empty    ;==== виведення повідомдення про пусту чергу
  (if (and (eq? (front q) (rear q))  (eq? '() (cdr (front q)))   )
   (begin
    (set! x (car (front q)))
```

```
 (set-car! (front q) '() )
 x )
(begin
 (set! x (car (front q)))
 (set-car! q (cdr (front q)) )
 x ))))
```

Зміст звіту

1. Титульний лист
2. Мета роботи
3. Умова завдання
4. Аналіз задачі та математичні забезпечення для розв’язання (у випадку математичної задачі)
5. Обґрунтування вибору мови функціонального програмування та IDE
6. Код
7. Скріншот роботи та результату
8. Оцінка достовірності результату (перевірка правильності результату)
9. Висновок

Завдання до лабораторної роботи 6

1. **Написати процедури, що обробляють вектори відповідно до правил векторної алгебри**
2. **Написати процедури, що обробляють черги відповідно до правил FIFO (перший прийшов — перший пішов) та стеків за правилами LIFO (останній прийшов - перший пішов)**

Номер варіанта	Умова завдань
1.	1.1. Створити вектор. Визначити максимальне та мінімальне значення серед елементів із парними та непарними індексами. Вивести мінімальний, максимальний елементи та їх індекси.
	1.2. Побудувати стек натуральних чисел. Вивести на екран створений стек. Надрукувати в зворотному порядку числа стеку, пропускаючи кратні заданому користувачем числу.
2. і	2.1 Створити вектор. Обчислити суму елементів між максимальним та мінімальними значеннями вектору. Вивести на екран максимальний, мінімальний елементи масиву, їх індекси та шукану суму елементів
	2.2 Створити чергу з чисел. Розрахувати їх середнє арифметичне та середнє геометричне. Надрукувати вміст черги, визначити кількість її елементів.
3.	3.1. Створити два вектори. Знайти найменший серед тих елементів першого вектору, які співпадають із значеннями елементів другого вектору. Вивести на екран вектори, найменший елемент та його індекс.
	3.2. Створити чергу з символів. Вибрати з черги англійські символи та записати їх у нову чергу. Вивести на екран вміст усіх черг.
4.	4.1.Створити вектор цілих чисел. Побудувати новий вектор із елементів першого вектору, в якому спочатку стоять числа, що діляться тільки на 2, потім ті, що діляться на 2 та 3, потім ті, що діляться тільки на 3. Надрукувати вхідний та вихідний вектори.
	4.2. Створити чергу з натуральних чисел. Продублювати парні числа черги в іншу чергу. Вивести на екран вхідну чергу а чергу з парними числами.
5.	5.1. Створити вектор чисел Знайти найбільший серед від’ємних та найменший серед додатних елементів вектору. Вивести вектор, значення знайдених елементів та їх індекси.

	5.2. Створити дві черги. Об’єднати дві черги в одну, елементи якої вибрані по черзі з вхідних черг, наприклад, перший елемент першої черги, перший елемент другої, другий елемент першої черги, другий елемент другої і т.д..
6.	6.1. Створити вектор цілих чисел. Знайти в векторі всі прості числа, скопіювати їх в новий вектор та надрукувати його.
	6.2. На вокзалі працює k кас, проте черга до них одна. Коли усі каси вільні, перші k клієнтів з черги підходять до кас. Інші чекають своєї черги. Як тільки кого-небудь буде обслужено і відповідна каса звільниться, наступна людина з черги підходить до цієї каси. Так продовжується до тих пір, доки не буде обслужено усіх клієнтів. Визначте час, за який буде обслужено усіх клієнтів.
7.	7.1. Створити вектор чисел. Знайти найбільший серед елементів вектору з непарними індексами та найменший серед елементів вектору з парними індексами. Вивести на екран значення знайдених елементів та їх індекси.
	7.2. Створити чергу з символів. Створити стек з тих елементів черги, які є голосними латинськими літерами.
8.	8.1. Створити вектор чисел, значення яких може повторюватися. Вивести на екран кількість входжень до вектору значень кожного з його елементів, індекс та значення елемента, яке повторюється найбільшу кількість разів.
	8.2. Створити дві черги чисел, серед яких я додатні та від’ємні. Здійснити порівняння черг. Якщо черги однакові, здійснити інверсію одної з них, інакше утворити нову чергу з від’ємних елементів двох попередніх черг.
9.	9.1. Створити вектор чисел з додатними, від’ємними та нульовими елементами. Переставити елементи вектору так, щоб спочатку були розташовані всі від’ємні елементи, потім усі додатні елементи, потім усі нульові. Вивести вектор на екран
	9.2. Створити чергу з символів, серед яких зустрічаються символи ‘(, ‘) . Створити другу чергу, в яку включити символи, які розміщені між парою дужок Надрукувати вихідні черги.
10.	10.1 Створити вектор чисел, значення яких можуть повторюватися. Видалити з вектору всі числа, які повторюються більше двох разів. Надрукувати вектор до та після видалення елементів
	10.2. Створити чергу з натуральних чисел, Визначити, чи можна з кожної трійки чисел утворити прямокутний трикутник, вважаючи числа черги довжинами сторін трикутника. Якщо можна, обчислити та записати в нову чергу радіуси вписаних кіл за формулою: $r = \frac{a+b-c}{2}$, де *a, b* – довжини катетів прямокутного трикутника, *c* – довжина його гіпотенузи.
11.	11.1. Створити вектор чисел з додатними, від’ємними та нульовими елементами. Видалити з вектору всі нульові елементи. Решту елементів розмістити так, щоб додатні і від'ємні елементи чергувалися. Надрукувати вектор до та після переміни.
	11.2. У магазині стоїть черга з *n* покупців. Заданий час обслуговування покупця з черги у вигляді цілого числа в діапазоні від 1 до *t*1, та час додавання нового покупця до черги – це ціле число в діапазоні від 1 до *t*2. Промоделювати стан черги (тобто показати час виникнення подій - обслуговування та додавання покупця) за період часу T (T>t1, T>t2). Вивести на екран залишок черги.
12.	12.1. Створити вектор чисел. Відсортувати вектор за методом бульбашкового сортування. Здійснити пошук заданого користувачем елемента масиву за методом бінарного пошуку.
	12.2. Створити чергу з цілими числами. Вибрати з черги елементи, які є кратними заданому числу і надрукувати їх. Додати до черги парні числа. Кількість елементів, що потрібно додати або вибрати визначає користувач.
13.	13.1. Створити два вектори чисел. Визначити та вивести на екран скалярний добуток двох векторів та косинус кута між ними.

	13.2. Створити чергу елементами якої є назви задач та їх пріоритети. Створити нову чергу, в якій задачі упорядковані за пріоритетами. Вивести на екран відсортовану чергу.
14.	14.1. Створити вектор чисел. Знайти найбільший серед від'ємних та найменший серед додатних елементів вектору. Вивести вектор, значення знайдених елементів та їх індекси.
	14.2. Створити стек натуральних чисел. Вибрати зі стеку числа, які не повторюються і записати їх у чергу. Надрукувати стек і чергу вибраних елементів
15.	15.1. Створити вектор чисел. Вивести на екран елементи, які менші за значення максимального і більше за значення мінімального елементів.
	15.2. Створити дві черги з різною кількістю елементів. Об'єднати черги, включивши в об'єднану чергу однакову кількість елементів з кожної вхідної черги.
16.	16.1. Створити два вектори цілих чисел, Побудувати третій вектор, в якому кожний елемент дорівнює найбільшому спільному дільнику (НСД) відповідних елементів вхідних векторів. Ввести на екран вектор НСД
	16.2. Створити чергу з символів. Із символів черги утворити стек із заданої користувачем кількості елементів. Об'єднати стек і чергу так, щоб утворити симетричну чергу.
17.	17.1. Створити вектор чисел. Визначити, скільки в цьому векторі елементів, які більше двох своїх сусідів справа та зліва. Вивести такі елементи, їх індекси та порахувати їх кількість. Крайні елементи вектору не враховуються, оскільки у них недостатньо сусідів.
	17.2. Створити чергу з n латинських символів. Вибрати із черги рівну кількість голосних і приголосних і записати їх до нової черги. Надрукувати нову чергу
18.	18.1. Створити вектор чисел, значення елементів якого є додатними та від'ємними числами. Вивести на екран суму елементів, розташованих до першого від'ємного елементу та добуток елементів, розташованих після останнього від'ємного елемента. Якщо від'ємних елементів немає, то вивести відповідне повідомлення.
	18.2. Створити чергу з *n* символів. Якщо черга симетрична, то вивести 'YES, інакше 'NO. Черга симетрична, якщо i-й символ з початку, співпадає з n-i+1 символом з кінця. У випадку несиметричної черги вивести на екран перші незбіжні елементи.
19.	19.1. Створити вектор чисел. Відсортувати вектор за алгоритмом Шелла (Shell sort) та здійснити пошук в масиві за алгоритмом рекурсивного лінійного пошуку
	19.2. Створити чергу з n латинських символів. Вибрати із черги голосні та приголосні символи і утворити з них слово. Надрукувати отримане слово.
20.	20.1. Створити вектор чисел. Знайти серед елементів вектору ті, які є числами Фібоначчі, факторіалами або квадратами числа.
	20.2. Створити чергу з натуральних чисел, Визначити, чи можна з кожної трійки чисел утворити рівностороннього трикутник, вважаючи числа черги довжинами сторін трикутника. Якщо можна, обчислити та записати в нову чергу радіуси описаних кіл за формулою: $r = \frac{a\sqrt{3}}{3}$, де a – це довжина сторони рівностороннього трикутника.

Лабораторна робота 7
Обробка рядків та файлів мовами функціонального програмування

Мета

Опанувати теоретичні основи обробки рядків та текстових файлів мовами функціонального програмування та розробити програми їх реалізації

Рейтингова таблиця лабораторної роботи 7

Вид роботи	Кількість балів	Deadline
Код задачі 1	4	листопад
Звіт	1	
Якість		
Разом	**5**	

Теоретичні відомості

Таблиця 7_1. Процедури обробки рядків мовами SCHEME, RACKET

Ім'я процедури	Призначення
(string? *obj*)	Повертає #t, якщо obj є рядком, інакше повертає #f.
(make-string *k*)	Повертає щойно виділений рядок довжиною k.
(make-string *k char*)	Всі елементи рядка ініціалізуються символом char
(string-length *string*)	Повертає кількість символів у вказаному рядку
(string-ref *string k*)	Повертає k-й символ рядка з використанням індексації нульового походження.
(string-set! *string k char*)	Зберігає char в k-му елементі рядка і повертає невизначене значення.
(string=? $string_1$ $string_2$)	Повертає #t, якщо два рядки однакової довжини і містять однакові символи в однакових позиціях, інакше повертає #f.
(string-ci=? string1 string2)	Повертає #t, якщо рядки відрізняються тільки регістром
(string<? $string_1$ $string_2$)	Порівняння рядків
(string>? $string_1$ $string_2$)	Порівняння рядків
(string<=? $string_1$ $string_2$)	Порівняння рядків
(string>=? $string_1$ $string_2$)	Порівняння рядків
(string-ci<? string1 string2)	Порівняння рядків
(string-ci>? string1 string2)	Порівняння рядків
(string-ci<=? string1 string2)	Порівняння рядків
(string-ci>=? string1 string2)	Порівняння рядків
(substring string start end)	Повертає підрядок рядка з позиції start до позиції end
(string-append string ...)	Повертає рядок, символи якого утворюють конкатенацію даних рядків.

(string->list string)	Повертає список символів
(list->string list)	Повертає рядок, сформований із списку символів
(string-copy string)	Повертає копію заданого рядка.
(string-fill! string char)	Зберігає char у кожному елементі заданого рядка

Приклад коду обробки рядків

```
;Записати кожне речення тексту в порядку зростання кількості голосних букв в
;слові.
(define T (quote("hello mrs." "aaaaaaaaa" "what are you doing?" "bbbbbbbbbbb"
"good bye.")))
;============= визначити голосні ==================
(define (isVowelChar char)
  (cond ((eq? char #\e) 1)
        ((eq? char #\y) 1)
        ((eq? char #\u) 1)
        ((eq? char #\i) 1)
        ((eq? char #\o) 1)
        ((eq? char #\a) 1)
        (else 0)
  ))
;===========порахувати кількість голосних ==============
 (define (vowelCount sentence)
  (if(> (length sentence) 0)
  (+ (isVowelChar (car sentence)) (vowelCount(cdr sentence)))
  0)
 )
;===============знайти слово з максимальною кількістю голосних==
(define (maxVowelSentence word sentence)
  (if (not(null? sentence))
      (if(> (vowelCount(string->list word)) (vowelCount(string->list (car
sentence))))
         (maxVowelSentence word (cdr sentence))
         (maxVowelSentence (car sentence) (cdr sentence))
         )
      word
  )
 )
;==========видалити елемент зі списку==================
(define delete
  (lambda (item list)
    (cond
     ((equal? item (car list)) (cdr list))
     (else (cons (car list) (delete item (cdr list)))))
))
;===========сортування слів по кількооті голосних=====================
(define (sortText text)
  (if (not(null? text))
      (cons (maxVowelSentence (car text) (cdr text))
            (sortText (delete (maxVowelSentence (car text) (cdr text)) text)))
  )
)
(sortText T) ; виклик процедур
```

Приклади коду обробки текстових файлів

```
;====== запис рядків у файл==============
(let ((port (open-output-file "f:\\kovalyuk500\\!KNU_Shevchenka\\f02")))
; в імені не припустимі пробіли та кирилицю
(write '(When I find myself in times of trouble) port)
(write '(Mother Mary comes to me) port)
(write '(Speaking words of wisdom, "Let it be") port)
(close-output-port port))
```

```
; ====== результат файлу в Блокноте============
;(when i find myself in times of trouble)(mother mary comes to me)(speaking words
of wisdom ;(unquote "Let it be")
;==============ВИВЕДЕННЯ файлу на екран =================
(define in (open-input-file "f:\\kovalyuk500\\!KNU_Shevchenka\\f02"))
(read in)
(read in)
(read in)
(close-input-port in)

;=========виведення рядків файлу в заданій послідовності========
(define (get-song n)  ; вибрати номер рядка файлу
  (let ((port (open-input-file "f:\\kovalyuk500\\!KNU_Shevchenka\\f02")))  ;
відкрити файл
    (skip-songs (- n 1) port)  ; пропустити рядок
    (let ((answer (read port)))
      (close-input-port port)
      answer)))

(define (skip-songs n port)   ; пропустити рядки
  (if (= n 0)
      'done
      (begin (read port)
             (skip-songs (- n 1) port))))

(display "Using a File as a Database")
(newline)
(get-song 2)
(get-song 1)
(get-song 3
(get-song 5)
;===============виведення файлу на екран як в блокноті======================
(define (print-file name)
  (let ((port (open-input-file name))) ;відкрити порт
    (print-file-helper port)           ;вивести рядки файлу
    (close-input-port port)            ; закрити порт
    'done))

(define (print-file-helper port)     ; допоміжна процедура друку файлу
  (let ((action (read port)))         ; читати порт
      (if (eof-object? action)        ;якщо кінець файлу
        'done                         ; робота звкінчена
        (begin (write action)         ; інакше писати файл
               (print-file-helper port)))))

(display "printed source file") ; виклик процедур
(newline)
(print-file "f:\\kovalyuk500\\!KNU_Shevchenka\\f02")
;==============================
```

Зміст звіту

1. Титульний лист
2. Мета роботи
3. Умова завдання
4. Аналіз задачі та математичні забезпечення для розв’язання (у випадку математичної задачі)
5. Обґрунтування вибору мови функціонального програмування та IDE
6. Код
7. Скріншот роботи та результату
8. Оцінка достовірності результату (перевірка правильності результату)
9. Висновок

Завдання до лабораторної роботи 7

1. Написати процедури, що обробляють рядки, які зчитані з текстових файлів. Результати обробки рядків записати до текстових файлів

Номер варіанта	Умова завдань
1.	Записати в текстовий файл n рядків тексту, що задаються програмою на функціональній мові програмування. Зчитати рядки із створеного програмою файлу, вивести їх на екран. Закодувати кожне друге слово в парному рядку за шифром Цезаря. У цьому шифрі кожна буква тексту замінюється на іншу, яка знаходиться на фіксовану кількість букв далі в алфавіті. Наприклад, якщо p_i літера слова, k – ключ слова (зсув літер алфавіту на k позицій), то закодована літера визначається як $c_i = p_i+k$. Записати в новий текстовий файл зашифрований текст
2.	Записати в текстовий файл n рядків тексту, що задаються програмою на функціональній мові програмування. Зчитати рядки із створеного програмою файлу, вивести їх на екран. Порахувати кількість повторень кожної літери в рядках тексту. Замінити задану користувачем літеру в тексті на її порядковий номер в алфавіті. Записати в новий текстовий файл результат обробки тексту
3.	Записати в текстовий файл n рядків тексту, що задаються програмою на функціональній мові програмування. Зчитати рядки із створеного програмою файлу, вивести їх на екран. Замінити в кожному реченні всі входження заданого слова на задане нове слово Записати в новий текстовий файл результат обробки тексту
4.	Записати в текстовий файл n рядків тексту, що задаються програмою на функціональній мові програмування. Зчитати рядки із створеного програмою файлу, вивести їх на екран. Зробити великою усі літери кожного слова, яке починається з великої літери. Записати в новий текстовий файл результат обробки тексту.
5.	Записати в текстовий файл n речень тексту, що задаються програмою на функціональній мові програмування. Речення закінчується символом «точка». Зчитати рядки із створеного програмою файлу, вивести їх на екран. Якщо перше слово речення починається з маленької літери, то замінити її на велику. Записати в новий текстовий файл результат обробки тексту
6.	Записати в текстовий файл n рядків тексту, що задаються програмою на функціональній мові програмування. Зчитати рядки із створеного програмою файлу, вивести їх на екран. В початок кожного рядка тексту вставити задане користувачем слово. Записати в новий текстовий файл результат обробки тексту.
7.	Записати в текстовий файл n речень тексту, що задаються програмою на функціональній мові програмування. Речення – це рядок, що закінчується символом «точка». Зчитати рядки із створеного програмою файлу, вивести їх на екран. Переписати кожне речення, розташувавши слова в порядку зворотному до алфавітного Записати в новий текстовий файл результат обробки тексту
8.	Записати в текстовий файл n речень тексту, що задаються програмою на функціональній мові програмування. Речення – це рядок, що закінчується символом «точка». Зчитати рядки із створеного програмою файлу, вивести їх на екран. Переписати кожне речення, розташувавши слова в порядку зменшення довжини слів. Записати в новий текстовий файл результат обробки тексту
9.	Записати в текстовий файл n рядків тексту, що задаються програмою на функціональній мові програмування. Зчитати рядки із створеного програмою файлу, вивести їх на екран. У кожному слові зчитаного тексту поміняти місцями першу та останню літери. Записати в новий текстовий файл результат обробки тексту
10.	Записати в текстовий файл n рядків тексту, що задаються програмою на функціональній мові програмування. Зчитати рядки із створеного програмою файлу, вивести їх на екран. Побудувати список пар: (<слово> <частота повторення в тексті>). Записати в новий текстовий файл побудовані пари.

11.	Записати в текстовий файл n рядків тексту, що задаються програмою на функціональній мові програмування. Зчитати рядки із створеного програмою файлу, вивести їх на екран. Видалити в початковому тексті з кожного слова його закінчення відповідно до заданого словника. Словник закінчень представляти списком рядків. Записати в новий текстовий файл результат обробки тексту
12.	Записати в текстовий файл n рядків тексту, що задаються програмою на функціональній мові програмування. Зчитати рядки із створеного програмою файлу, вивести їх на екран. Визначити однакові слова в кожному рядку, надрукувати їх та їх кількість. Записати в новий текстовий файл однакові слова кожного рядка.
13.	Записати в текстовий файл 2 рядки тексту, що задаються програмою на функціональній мові програмування. Зчитати рядки із створеного програмою файлу, вивести їх на екран. у першому рядку кожне слово на парній позиції замінити словом на непарній позиції другого рядка. Записати в новий текстовий файл результат обробки тексту
14.	Записати в текстовий файл n рядків тексту, що задаються програмою на функціональній мові програмування. Зчитати рядки із створеного програмою файлу, вивести їх на екран. Замінити слово з найбільшою кількістю голосних літер на послідовність цифр відповідно до довжині слова. Наприклад, знайдене слово "moloko", замінене слово "123456". Записати в новий текстовий файл результат обробки тексту.
15.	Записати в текстовий файл n рядків тексту, що задаються програмою на функціональній мові програмування. Зчитати рядки із створеного програмою файлу, вивести їх на екран. Видалити з тексу усі стоп слова. Записати в новий текстовий файл результат обробки тексту. Словник стоп слів поданий https://countwordsfree.com/stopwords
16.	Записати в текстовий файл n рядків тексту, що задаються програмою на функціональній мові програмування. Зчитати рядки із створеного програмою файлу, вивести їх на екран. В рядках можуть зустрічатися цифрові символи. Порахувати кількість цифрових символів і записати їх в новий текстовий файл
17.	Записати в текстовий файл n рядків тексту, що задаються програмою на функціональній мові програмування. Зчитати рядки із створеного програмою файлу, вивести їх на екран. Підрахувати кількість слів, які містять однакову кількість голосних і приголосних літер. Записати слова з однаковою кількістю голосних і приголосних літер в новий текстовий файл.
18.	Записати в текстовий файл рядок алфавітних, цифрових символів, довільну кількість символів '(' та ')', що задається програмою на функціональній мові програмування. Вивести на екран усі символи, розташовані всередині пари дужок, і підрахувати їх кількість. Записати в новий текстовий файл символи, розташовані всередині пари дужок.
19.	Записати в текстовий файл n рядків тексту, що задаються програмою на функціональній мові програмування. Зчитати рядки із створеного програмою файлу, вивести їх на екран. Визначити найдовшу послідовність однакових символів в рядках, що йдуть поспіль без розділових символів. Записати в новий текстовий файл знайдену послідовність.
20.	Записати в текстовий файл n рядків тексту, що задаються програмою на функціональній мові програмування. Перевірити, чи є введений рядок ідентифікатором змінної відповідно до правил написання ідентифікаторів в алгоритмічних мовах програмування. Зчитати рядки із створеного програмою файлу, вивести їх на екран. Записати в новий текстовий файл слова, які можуть бути ідентифікаторами змінних.

Лабораторна робота 8
Символьні обчислення мовами функціонального програмування. Операції з многочленами.

Мета

Опанувати теоретичні основи символьних обчислень мовами функціонального програмування. Розробити програми символьного диференціювання та реалізації арифметики поліномів

Рейтингова таблиця лабораторної роботи 8

Вид роботи	Кількість балів	Deadline
Код задачі 1	5	Грудень (до 10.12.2020)
Звіт	1	
Якість		
Разом	**6**	

Теоретичні відомості

Правила диференціювання

Диференціювати будь-який вираз можна, застосовуючи такі правила редукції (1) – (4):

(1) $\frac{dc}{dx} = 0$ для константи або змінної, що не є x

(2) $\frac{dx}{dx} = 1$

(3) $\frac{d(u+v)}{dx} = \frac{du}{dx} + \frac{dv}{dx}$

(4) $\frac{d(uv)}{dx} = u(\frac{dv}{dx}) + v(\frac{du}{dx})$

(5) $\frac{d(u^n)}{dx} = nu^{n-1}\left(\frac{du}{dx}\right)$

1. Правила (3) та (4) по суті своїй рекурсивні.
2. Щоб отримати похідну суми, спочатку потрібно отримати похідні доданків і їх скласти.
3. Кожний доданок в свою чергу може бути виразом, який потрібно розкласти на складові.
4. Розбиваючи їх на все більш дрібні частини, можна дійти до стадії, коли всі частини є або **константами**, або **змінними**, і їх похідні дорівнюватимуть або **0**, або **1**.

Застосування загальних правил диференціювання відносно позначень, d / dx [форма], де *x* - змінна, щодо якої береться похідна, а форма - формула, похідна від якої потрібна.

1. d/dx[c] = 0 , where c is a numeric constant.
2. d/dx[x] = 1
3. d/dx[v] = 0 , where v is a variable other than x.
4. d/dx[u + v] = d/dx[u] + d/dx[v]
5. d/dx[u - v] = d/dx[u] - d/dx[v]
6. d/dx[-v] = - d/dx[v]
7. d/dx[u * v] = u * d/dx[v] + v * d/dx[u]
8. d/dx[u / v] = (v * d/dx[u] - u * d/dx[v]) / v^2
9. d/dx[u^c] = c * $u^{(c - 1)}$ * d/dx[u] , where c is constant.
10. d/dx[sqrt(u)] = (1/2) * d/dx[u] / sqrt(u)
11. d/dx[log(u)] = (d/dx[u]) / u
12. d/dx[exp(u)] = exp(u) * d/dx[u]

13. d/dx[sin(u)] = cos(u) * d/dx[u]
14. d/dx[cos(u)] = - sin(u) * d/dx[u]
15. d/dx[tan(u)] = (1 + tan(u)^2) * d/dx[u]

Арифметика поліномів (многочленів)

Многочлени з однією змінною можна ділити один на одного, отримуючи частку і залишок. Для ділення розділимо старший член діленого на старший член дільника. В результаті вийде перший терм частки. Потім помножимо результат на дільник, віднімемо многочлен, що вийшов з діленого і, рекурсивно ділячи різницю на дільник, отримаємо решту частки. Зупиняємося, коли порядок дільника перевищить порядок ділимого, і оголошуємо залишком те, що тоді буде називатися діленим. Крім того, якщо коли-небудь ділене виявиться нулем, повертаємо нуль як і частку, і залишок.

Додавання многочленів відбувається по термах. При цьому породжується новий терм того самого порядку, в якому коефіцієнт є сумою коефіцієнтів доданків. Терми одного доданка, для яких немає відповідності в іншому, просто додаються до породжуваного многочлену-сумі.

Щоб перемножити два списки термів, треба кожен терм з першого списку помножити на всі терми другого. Утворені списки термів накопичуються і утворюють суму. Множення двох термів дає терм, порядок якого дорівнює сумі порядків множників, а коефіцієнт дорівнює добутку коефіцієнтів множників.

Умови зростання та спадання функції

1. Необхідною і достатньою умовою зростання функції *f(x)*, що диференцується на інтервалі (*a; b*), є умова

$$f'(x) \geqslant 0, \quad x \in (a; b),$$

2. Необхідною і достатньою умовою зменшення функції f (x), що диференцується на інтервалі (a; b), є умова

$$f'(x) \leqslant 0, \quad x \in (a; b).$$

3. Точка, в якій перша похідна змінює знак плюс на мінус, є точкою максимуму.

$$f'(x) > 0 \quad \text{при} \quad x < x_0 \quad , \quad f'(x) < 0 \quad \text{при} \quad x > x_0.$$

4. Точка, в якій перша похідна змінює знак мінус на плюс, є точкою мінімуму.

$$f'(x) < 0 \quad \text{при} \quad x < x_0 \quad \text{и} \quad f'(x) > 0 \quad \text{при} \quad x > x_0,$$

5. Точка, в якій перша похідна дорівнює нулю, а друга похідна від'ємна, є точкою максимуму

$$f'(x_0) = 0, \; f''(x_0) < 0$$

6. . Точка, в якій перша похідна дорівнює нулю, а друга похідна додатна, є точкою мінімуму

$$f'(x_0) = 0, \; f''(x_0) > 0.$$

7. Умова опуклості донизу функції: *f(x)* на інтервалі (*a; b*) – друга похідна невід'ємна:

$$f''(x) \geqslant 0, \quad x \in (a; b).$$

8. Необхідною і достатньою умовою опуклості вгору функції *f(x)* на інтервалі (*a; b*) є умова

$$f''(x) \leqslant 0, \quad x \in (a; b),$$

9. Умова наявності точки перегину є:

$$f''(x_0) = f'''(x_0) = \ldots = f^{(n-1)}(x_0) = 0, \quad \text{а} \quad f^{(n)}(x_0) \neq 0;$$

якщо n - непарне число, то x_0 є точкою перегину, якщо n - парне число, то x_0 не є точкою перегину.

Приклад коду символьного обчислення

```
;========= символьне диференціювання====
(define (variable? x) (symbol? x))
(define (same-variable? v1 v2)
  (and (variable? v1) (variable? v2) (eq? v1 v2)))
(define (make-sum a1 a2) (list '+ a1 a2))
(define (make-product m1 m2) (list '* m1 m2))
(define (sum? x)
  (and (pair? x) (eq? (car x) '+)))
(define (addend s) (cadr s))
(define (augend s) (caddr s))
(define (product? x)
  (and (pair? x) (eq? (car x) '*)))
(define (multiplier p) (cadr p))
(define (multiplicand p) (caddr p))

(define (deriv exp var)
  (cond ((number? exp) 0)
    ((variable? exp)
     (if (same-variable? exp var) 1 0))
    ((sum? exp)
     (make-sum (deriv (addend exp) var)
               (deriv (augend exp) var)))
    ((product? exp)
     (make-sum
       (make-product (multiplier exp)
                     (deriv (multiplicand exp) var))
       (make-product (deriv (multiplier exp) var)
                     (multiplicand exp))))
    (else
     (display "unknown expression type - DERIV" ))))

(deriv '(* (* x y) (+ x 3)) 'x)
(deriv '3 'x)
(deriv '(* x y) 'x)
(deriv '(! x y) 'x)

;================== додавання поліномів===============
(define (add-polynomial a b)
  (define (add-polynomial-aux a b)
    (cond ((and (null? a) (null? b))( list))
          ((null? a) b)
          ((null? b) a)
          (else(cons
                (+ (car a) (car b))
                (add-polynomial-aux (cdr a) (cdr b))))))
  (reverse (add-polynomial-aux (reverse a)
                               (reverse b))))

(define a (list 3 0 -2 1 -1))
(define b (list 1 2 3 4))
(add-polynomial a b)
```

Зміст звіту

1. Титульний лист
2. Мета роботи
3. Умова завдання
4. Аналіз задачі та математичні забезпечення для розв’язання (у випадку математичної задачі)
5. Обґрунтування вибору мови функціонального програмування функціонального програмування та IDE
6. Код

7. Скріншот роботи та результату
8. Оцінка достовірності результату (перевірка правильності результату)
9. Висновок

Завдання до лабораторної роботи 8

1. Написати процедури, що здійснюють символьне диференціювання та реалізують арифметику поліномів Результати записати до текстових файлів

Номер варіанта	Умова завдань
1.	Розробити процедури для обчислення суми, різниці та добутку двох многочленів (поліномів) від одної змінної, заданих списком своїх коефіцієнтів та степенями у вигляді: $P(x)=a_0x^n+a_1x^{n-1}+...+a_{n-1}x+a_n$ та $R(x)=b_0x^m+a_1x^{m-1}+...+a_{m-1}x+a_m$ Вивести на екран вирази, що є результатом суми та добутку двох многочленів. Врахувати, що многочлени можуть бути задані з різними степенями. Значення многочленів при заданому користувачем значенні змінної не обчислювати. Результат подати в символьному вигляді.
2.	Розробити процедури для обчислення добутку, частки та остачі від ділення двох многочленів (поліномів) від одної змінної, заданих списком своїх коефіцієнтів та степенями у вигляді: $P(x)=a_0x^n+a_1x^{n-1}+...+a_{n-1}x+a_n$ та $R(x)=b_0x^m+a_1x^{m-1}+...+a_{m-1}x+a_m$ Вивести на екран вирази, що є результатом множення та ділення двох многочленів. Врахувати, що многочлени можуть бути задані різними з степенями. Значення многочленів при заданому користувачем значенні змінної не обчислювати. Результат подати в символьному вигляді.
3.	Розробити процедури для розкладання многочлена на множники відповідно до правил алгебри. Многочлен заданий списком своїх коефіцієнтів та степенем від одної змінної у вигляді: $P(x)=a_0x^n+a_1x^{n-1}+...+a_{n-1}x+a_n$ Вивести на екран вирази, що є результатом розкладання многочлена. Окремою процедурою перевірити правильність розкладання многочлена на множники, здійснивши множення отриманих множників.
4.	Розробити процедури для визначення коренів *n*-ого степеню алгебраїчного рівняння вимірності n>3 від одної змінної виду $a_0x^n+a_1x^{n-1}+...+a_{n-1}x+a_n=0$. Алгебраїчне рівняння задавати списком своїх коефіцієнтів та степенем. Ввести на екран символьні вирази, що є коренем рівняння алгебраїчного рівняння степені n>3. Значення многочленів при заданому користувачем значенні змінної не обчислювати.
5.	Розробити процедуру обчислення найбільшого спільного дільника (НСД) двох многочленів, заданих списком своїх коефіцієнтів та степенями у вигляді $P(x)=a_0x^n+a_1x^{n-1}+...+a_{n-1}x+a_n$ та $R(x)=b_0x^m+a_1x^{m-1}+...+a_{m-1}x+a_m$. Результатом має бути многочлен у вигляді $f(x)=c_0x^k+c_1x^{k-1}+...+c_{k-1}x+c_k$ Значення многочленів при заданому користувачем значенні змінної не обчислювати. Результат подати в символьному вигляді. У разі відсутності НСД вивести відповідне повідомлення.
6.	Розробити процедури для розкладання многочлена по степенях двочлена. Многочлен та двочлен задані списком своїх коефіцієнтів та степенями у вигляді $f(x)=a_0x^n+a_1x^{n-1}+...+a_{n-1}x+a_n$ та $g(x)=b_0x+b_1$. Для спрощення можна використовувати схему Горнера [3.12]. Вивести на екран многочлен, який є результатом розкладання, у вигляді $P(x)=c_ng(x)^n+c_{n-1}g(x)^{n-1}+...+c_1g(x)+c_0$. Значення многочленів при заданому користувачем значенні змінної не обчислювати. Результат подати в символьному вигляді.
7.	Нехай P1, P2 та P3 – многочлени, задані виразами виду $P(x)=a_0x^n+a_1x^{n-1}+...+a_{n-1}x+a_n$. Нехай Q1 є добутком P1 и P2, а Q2 - добутком P1 и P3. Написати програму для обчислення найбільшого спільного дільника Q1 та Q2. .

	Значення многочленів при заданому користувачем значенні змінної не обчислювати. Результат подати в символьному вигляді. Многочлени задавати списком своїх коефіцієнтів та степенями.
8.	Написати програму символьного диференціювання по одній змінній алгебраїчних виразів, які містять усі арифметичні операції (+, –, *, /) та тригонометричні функції sin(x), cos(x), tg(x), ctg(x). Результат подати у формі алгебраїчного виразу. Значення алгебраїчного виразу при заданому користувачем значенні змінної не обчислювати.
9.	Написати програму символьного диференціювання по одній змінній алгебраїчних виразів, які містять усі арифметичні операції (+, –, *, /) та тригонометричні функції sh(x),, ch(x), arctg(x), arcsin(x). Результат подати у формі алгебраїчного виразу. Значення алгебраїчного виразу при заданому користувачем значенні змінної не обчислювати.
10.	Написати програму символьного диференціювання по одній змінній алгебраїчних виразів, які містять усі арифметичні операції (+, –, *, /), експоненціальні, показові та логарифмічні функції exp(x), pow (x, n), lg(x). Результат подати у формі алгебраїчного виразу. Значення алгебраїчного виразу при заданому користувачем значенні змінної не обчислювати.
11.	Написати програму символьного диференціювання по одній змінній поліному, який заданий списком своїх коефіцієнтів та степенем у вигляді: $f(x) = a_0 x^n + a_1 x^{n-1} + ... + a_{n-1} x + a_n$. Знайти першу, другу та третю похідні. Результати вивести у вигляді символьного виразу. Значення поліному та його похідних при заданому користувачем значенні змінної не обчислювати.
12.	Написати програму, яку досліджує на екстремум довільну функцію, яка задана алгебраїчним виразом, що містить усі арифметичні операції (+, –, *, /) та зведення у степінь. Для пошуку екстремумів використати першу та другу похідні від функції одної змінної. Результат вивести у вигляді виразів, що є першою та другою похідними, та вказати діапазони, в яких похідні міняють знак.
13.	Написати програму, яку визначає інтервали опуклості та вгнутості довільної функції, яка задана алгебраїчним виразом, що містить усі арифметичні операції (+, –, *, /), зведення у степінь та тригонометричні функції sin(x), cos(x). Для пошуку інтервалів опуклості та вгнутості використати першу та другу похідні від функції одної змінної.
14.	Написати програму, яку визначає інтервали зростання та спадання функції, яка задана алгебраїчним виразом, що містить усі арифметичні операції (+, –, *, /), зведення у степінь та тригонометричні функції tg(x), ctg(x). Для пошуку інтервалів зростання та спадання функції використати першу та другу похідні від функції одної змінної.
15.	Написати програму, яка для трьох многочленів, які задані списком своїх коефіцієнтів та степенями у вигляді $f(x) = a_0 x^n + a_1 x^{n-1} + ... + a_{n-1} x + a_n$. $g(x) = b_0 x^m + a_1 x^{m-1} + ... + a_{m-1} x + a_m$ та $h(x) = c_0 x^k + c_1 x^{k-1} + ... + c_{k-1} x + c_k$, перевіряє властивості асоціативного додавання $f(x)+(g(x)+h(x))=(g(x)+f(x))+h(x)$ та дистрибутивності $(g(x)+h(x))*f(x) = g(x)*f(x)+h(x)*f(x)$. Вивести на екран рядок “YES” або “NO” в залежності від виконання відповідних властивостей.
16.	Написати програму, яка визначає довільні коефіцієнти та будує многочлен виду $f(x) = x^n + a_1 x^{n-1} + ... + a_{n-1} x + a_n$ мінімального степеню зі старшим коефіцієнтом, рівним 1, у якого відомі прості корені та корені кратності 2. Для визначення коефіцієнтів многочлена використати формулу Вієта. Значення простих коренів та коренів кратності 2 задає користувач (може підбирати програма). Надрукувати многочлен з шуканими коефіцієнтами.
17.	Написати програму пошуку коренів многочлена, який заданий списком своїх коефіцієнтів та степенем у вигляді $P(x) = a_0 x^n + a_1 x^{n-1} + ... + a_{n-1} x + a_n$. Якщо α є коренем многочлена $P(x)$, тоді лінійний поліном $(x-\alpha)$ ділить $P(x)$, тобто, існує інший многочлен $Q(x)$ такий, що $P(x)= (x -\alpha)\times Q(x)$. Якщо корені повторюються, то порахувати їх кратність

18.	Написати програму, яка досліджує довільну функцію *y = f(x)*, що задана параметрично з параметром *t*, на екстремум. Для цього визначити перші похідні y'_x, y'_t та x'_t, критичні точки функції, перевірити знаки похідної y'_x при переході через критичні точки, визначити значення функції в критичних точках.
19.	Реалізуйте процедуру (factorize) для виконання розкладання многочленів виду a^2-b^2, a^3-b^3 и a^3+b^3 за формулами скороченого множення. Процедура приймає один аргумент – вираз, який потрібно розкласти на множники. Піднесення до степеню в початкових виразах реалізувати за допомогою вбудованої процедури expt. Приклади виклику процедури: (factorize '(- (expt x 2) (expt y 2))) ⇒ (* (- x y) (+ x y))
20.	Знайти корені лінійного диференціального рівняння першого порядку виду $y' + a(x)y = f(x)$ будь-яким методом. Загальне рішення диференціального рівняння першого порядку виражено у вигляді $y = \frac{\int u(x) f(x)dx + C}{u(x)}$, де *C* – довільна константа, *u(x)* – інтегруючий множник, що визначається за формулою $u(x) = \exp(\int a(x)dx)$, *f(x)* – довільна функція

Посилання на джерела до лабораторної роботи №8

1. Многочлены. Режим доступу: http://www.uic.unn.ru/~zny/algebra/lectures/lectures/05_Polynomials.pdf
2. Розкладання многочленів на множники. Режим доступу: https://disted.edu.vn.ua/courses/learn/1144
3. Алгебраические уравнения (стр. 146). Режим доступу: http://mmf.pskgu.ru/ebooks/gusak/gusak_gl08.pdf
4. Корни полинома. Режим доступу: https://www.matburo.ru/Examples/Files/LinAlg35.pdf
5. Наибольший общий делитель. Алгоритм Евклида (приклад стор. 8). Режим доступу:http://www.uic.unn.ru/~zny/algebra/lectures/lectures/05_Polynomials.pdf
6. Розв'язок алгебраїчних рівнянь з використанням схеми Горнера. Режим доступу http://www.mathros.net.ua/rozvjazok-algebraichnyh-rivnjan-metodom-poslidovnyh-nablyzhen-z-vykorystannjam-shemy-gornera.html
7. Кудрявцев Л Д, Кутасов А Д, Чехлов В И, Шабунин М И. Сборник задач по математическому анализу. Том 1. Предел. Непрерывность. Дифференцируемость. — М ФИЗМАТЛИТ, 2003. (стр. 366 Справочные сведения). Режим доступу http://math.sfu-kras.ru/sites/default/files/kudr_zad_v1.pdf
8. Формулы Виета (стр. 24). Режим доступу: http://www.uic.unn.ru/~zny/algebra/lectures/lectures/05_Polynomials.pdf