Практична робота №2
Знайомство з базовим синтаксисом мови програмування C#
Система оцінювання

№	Тема	К-ть балів
1.	Завдання 1	0,8
2.	Завдання 2	1
3.	Завдання 3	0,8
4.	Завдання 4	1
5.	Завдання 5	1
6.	*Здача звіту*	0,4
	Всього за практичну роботу	**5**
7.	Завдання 1 (ІНДЗ)	1
8.	Завдання 2 (ІНДЗ)	1
9.	Завдання 3 (ІНДЗ)	1
	Всього	**8**

Завдання на практичне заняття

1. З 1 липня 2018 року в Україні діє правило заокруглення готівкових розрахунків за умови відсутності монет відповідного номіналу в продавця чи покупця. Напишіть програму, яка зчитує ціни від користувача, поки не буде введено порожній рядок. Додаток повинен обчислити суму оплати в готівковій (округлити до найближчого 10-копієчника) та безготівковій формі. Одним із способів округлення є такий алгоритм дій:
    a. Визначити кількість копійок у сумарній вартості
    b. Обчислити остачу від ділення на 10.
    c. Округлити до меншого, якщо остача від ділення менша за 5. Інакше округлити до більшого.

2. ***(Операції з рядками)*** Препроцесинг тексту для подальшого машинного навчання на його основі передбачає підготовку цього тексту для використання відповідною моделлю навчання. Подібна обробка може включати багато етапів. Виконайте подібні кроки для тексту, зчитаного з консолі:
    a. Перетворіть всі літери в маленькі;
    b. Видаліть усі числа з тексту;
    c. Видаліть пунктуацію (символи !”#$%&’()*+,-./:;<=>?@[\]^_`{|}~) та незначущі пробіли;
    d. Виконайте токенізацію (перетворення тексту в набір слів);
    e. Видаліть стоп-слова (stopwords) з тексту. Масив таких слів можете визначити самостійно.

    Виведіть на екран отриману множину слів (bag of words).

3. ***(Регулярні вирази)*** Розробіть програмне забезпечення, яке буде здійснювати перевірку введеного тексту відповідно до варіанту (НомерУСпискуПідгрупи % 3 + 1):

Варіант 1	**Варіант 2**	**Варіант 3**
Перевіряти коректність введеної адреси електронної пошти. Для розуміння вирішення проблеми можете переглянути статтю від Microsoft.	Шукати веб-посилання в HTML-розмітці. Для розуміння вирішення проблеми можете переглянути статтю від Microsoft.	Заміняти дати в форматі ММ-ДД-РР на дати в форматі ДД-ММ-РР. Для розуміння вирішення проблеми можете переглянути статтю від Microsoft.

4. Рулетка має 38 комірок: 18 чорних, 18 червоних і 2 зелених. Зелені комірки позначаються як 0 та 00. Червоними комірками є 1, 3, 5, 7, 9, 12, 14, 16, 18, 19, 21, 23, 25, 27, 30 32, 34 та 36. Решта цілих чисел з діапазону від 1 до 36 є чорними.
   На рулетку можна зробити багато ставок. Для цієї вправи розглянемо наступні можливі ставки:
   • Одне число (від 1 до 36, 0 або 00)
   • Червоне vs Чорне
   • Парне vs Непарне (0 та 00 не включаються)
   • Від 1 до 18 vs від 19 до 36
   Напишіть програму, яка симулює кручення колеса рулетки за допомогою генератора випадкових чисел. Виведіть вибране число та всі ставки, які потрібно оплатити. Наприклад, для 13:
   На рулетці випало 13...
   Виплатити 13
   Виплатити Black
   Виплатити Odd
   Виплатити 1 to 18
   Якщо результати симуляції випадають на 0 або 00, слід вивести Виплатити 0 або Виплатити 00 без іншої інформації

5. Нижче наведена таблиця описує октави музичних нот, починаючи з middle C (C4) разом з їх частотами.

Нота	**Частота (Гц)**
C4	261,63
D4	293,66
E4	329,63
F4	349,23
G4	392,00
A4	440,00
B4	493,88

Програма має коректно виводити частоти для нот всіх октав – від C0 до C8. Для того, щоб їх знайти, потрібно враховувати наступне правило: частота ноти в октаві n – половина частоти ноти в октаві n+1. Додаток має підтримувати 2 варіанти роботи:

1) На вхід програми має надходити нота відповідної октави, а на виході виводитись її частота.
2) За введеною частотою визначити ноту.

Підказка: для виконання завдання потрібно виділити окремо символи з двосимвольної назви ноти. Далі можна визначити частоту ноти в четвертій октаві, а потім поділити її на 2^{4-x}, де x – номер октави, введеної користувачем.

ЗАВДАННЯ на ІНДЗ

1. Порівняйте продуктивність роботи різних циклів, доступних у мові програмування C#:
    a. while
    b. for
    c. foreach

    Створіть масив з 1000 000 цілих чисел, які генеруватимуться випадковим чином. Виконуйте додавання всіх елементів масиву за допомогою різних циклів. Для того, щоб засікти час виконання обходу, використовуйте клас StopWatch, приклад використання якого можна знайти тут. Порівняйте час роботи трьох циклів та зробіть висновок щодо їх використання.
2. На базі розробленого методу для заміру часу із задачі 1 дослідіть вплив наявності оператору try-catch на ситуації з генеруванням виняткової ситуації та без нього. Розгляньте 4 ситуації парсингу тексту з повтором операції 1000 000 разів:
    a. Парсинг без помилки (використання методу int.Parse() для рядка “1”);
    b. Парсинг з помилкою (використання методу int.Parse() для рядка “error”);
    c. Парсинг без помилки (використання методу int.TryParse() для рядка “1”);
    d. Парсинг з помилкою (використання методу int.TryParse() для рядка “error”);

    Опишіть зроблені висновки щодо використання оператору try-catch у звіті.
3. ***(Змінювані та незмінювані рядки)***. Порівняйте швидкодію та використання пам’яті при конкатенації змінюваних рядків (тип StringBuilder) та незмінюваних рядків (тип string). Для цього застосуйте цикли, що дописуватимуть до початкового рядка певним чином визначені символи 10000 разів. У якості прикладу для бенчмаркінгу перегляньте статтю.