ЗАВДАННЯ
Побудувати прототип інформаційно-довідкової системи про пересування містом трамваями у Львові

Мета завдання.

1. Отримати первинне ознайомленнями з деякими аспектами будови інформаційно-довідкових систем. Навчитись програмувати подібні системи.
2. Навчитись будувати сховища чи зображення даних для забезпечення роботи системи. Вміти проводити первинну підготовку даних до форматів, які забезпечують швидке реагування на запити користувачів і зручне програмування функцій інформаційно-довідкових систем.

Загальні кроки реалізації задачі.

1.Побудувати повний перелік назв трамвайних зупинок для всіх маршрутів в сукупності.
2.Обрати необхідні структури даних для зручного програмування і швидкого виконання на запити користувачів. Перенести в такі структури підготовлені дані.
3.Скласти список прогнозованих запитів до інформаційно-довідкової системи. Реалізувати запити як окремі функції.
4.Підготувати формати надсилання відповідей.

Схема реалізації.

1.Для коректного виконання наступних кроків проєкту треба мати в "письмовій" формі повний перелік назв всіх зупинок всіх трамвайних маршрутів в сукупності. Це може бути звичайний текстовий файл, в якому в кожному рядку записана назва однієї зупинки, наприклад:

```
Залізничний вокзал
Приміський вокзал
Площа Кропивницького
```

. . . . . і так далі. [Уважно з пробілами!!]

Первинна підготовка повного списку зупинок є досить надокучливою справою, але її треба виконати однократно. Дані можна взяти з файла "Scheme-Tram-and-Trolleybus.jpg". Подібні дані є, вірогідно, на різних сторінках сайту Львова, але немає певності, що дані вчасно оновлені.

Склавши загальний список, його треба сортувати за алфавітом, після чого вручну чи програмно викреслити назви, які повторюються.

Такий список потрібний для того, щоб забезпечити правильне однакове позначення в процесі програмування. Для перенесення назви зупинки в код програми можна просто копіювати і вставляти потрібну назву Ctrl-C, Ctrl-V. Якщо загального списку не буде, тоді при ручному друкуванні можливі такі випадки, як "Площа Кропивницького", "площа Кропивницького", "пл.Кропивницького", "Кропивницького", які з точки зору формального синтаксису означають різні назви.

Отже, підготовка списку сама по собі може бути окремим програмним субпроєктом.

2.Вибір і підготовка структур даних. Обмежимо наш проєкт вбудованими структурами даних: текст, список, кортеж, множина, словник.

Важливо добре продумати програмне зображення даних, викладене в п.1, з огляду на зручне програмування і швидкий пошук відповідей на запитання. Наприклад, можна взяти за основу словник, ключами якого будуть номери трамвайних маршрутів, а значеннями – три елементи: список зупинок в одному напрямі, список зупинок в зворотньому напрямі, множина всіх зупинок на маршруті. Наприклад:

```
tramway = {
 2: [
      [ "Коновальця", "Гординських", "Залізняка", "...",   \
        "Київська", "...", "Пасічна" ],   \
      [  ],   \
      {  }   \
    ],
 9: [ ["....."],
      [ "....." ],
      { "зупинка", "Київська", "Парк культури", "інші зупинки" }
    ]
 }

# програмна будова додаткових даних
# зворотній маршрут:
tramway[2][1] = tramway[2][0].copy();  tramway[2][1].reverse()
# перелік зупинок як множина:
tramway[2][2] = set(tramway[2][0])
# контроль
print(tramway[2][0], tramway[2][1], tramway[2][2], sep='\n')
```

Звернемо увагу, що достатньо записати перелік зупинок від початкової до кінцевої лише в одному напрямі. Решту можна обчислити програмно як показано.

```
# приклад: пошук спільних зупинок - операція над множинами
changeonesseat = tramway[2][2] & tramway[9][2]
print(changeonesseat)

# приклад: вибір напряму маршруту
fromstop = "Гординських"  # від цієї зупинки
tostop = "Київська"  # до такої зупинки
if tramway[2][0].index(fromstop) < tramway[2][0].index(tostop) :
  direction = tramway[2][0]
else: direction = tramway[2][1]
# контроль
print("напрям: ", direction)
```

Записане вище не є обов'язковою формою виконання завдання. Можна укласти інший спосіб зображення даних. Головна ідея вибору зображення полягає в тому, щоб підготувати наперед всі елементи програми так, щоб не обчислювати

багатократно одні й ті ж дані. Наприклад, для пошуку зупинок пересадки з одного маршруту в інший зручно використати просту операцію перетину множин зупинок, але зовсім не потрібно щоразу обчислювати ту саму множину для окремого маршруту. Для переліку зупинок під час руху трамвая краще переглядати списки зупинок в однаковий спосіб зліва направо, а для цього треба мати списки зупинок в одному напрямі і в зворотньому напрямі, інакше доведеться щоразу застосовувати функцію anylist.reverse(), що може збільшувати час обчислення відповідей на запити.

3-4.Приклади запитів і форматів відповідей. Для цілей нашого проєкту будемо будувати всі запити і відповіді в форматі назв трамвайних зупинок. Ключові слова запитів і відповідей підкреслені. Список ключових слів треба визначити окремо як частину інструкції користування інформаційно-довідковою системою.

Загальне зауваження. Якщо відповідь можна будувати різними способами, обрати будь-яку одну відповідь, щоб не ускладнювати проєкт.

3-4.1.
запит) Як потрапити від зупинки "Центр Довженка" на зупинку "Саксаганського"?
відповідь) Прийдіть на зупинку "Центр Довженка" трамваю 8. Сідайте на трамвай 8 в напрямку "Вернадського"-"Площа Соборна". Переїхати трамваєм 10 зупинок. Слухайте оголошення в трамваї про зупинки. ["Площа Соборна" є кінцевою зупинкою.]

3-4.2.
запит) Як потрапити від зупинки "Гайдамацька" на зупинку "Мечникова"?
відповідь) Прийдіть на зупинку "Гайдамацька" трамваю 9. Сідайте на трамвай 9 в напрямку "Торф'яна"-"Залізничний вокзал". Переїхати трамваєм 4 зупинки до "Підвальна". [Перейдіть на зупинку трамваю 2 "Руська". [Подумати, як зобразити в програмі сусідні зупинки]. Якщо перехід не потрібний, то пропустити цей пункт]. Сідайте (чи пересядьте) на трамвай 2 в напрямку "Коновальця"-"Пасічна". Переїхати трамваєм 4 зупинки. Слухайте оголошення в трамваї про зупинки.

3-4.3.
запит) Чи можна потрапити від зупинки "Академія мистецтв" на зупинку "Аквапарк"?
відповідь) Можна, з пересадкою з трамвая 4 на трамвай 3.

3-4.4.
запит) Скільки зупинок від зупинки "Погулянка" до зупинки "Магнус"?
відповідь) 11 зупинок без пересадки. [Якщо потрібна пересадка, повідомити про це].

3-4.5.
запит) Якими трамваями можна потрапити на зупинку "Головна пошта"?
відповідь) 1, 2

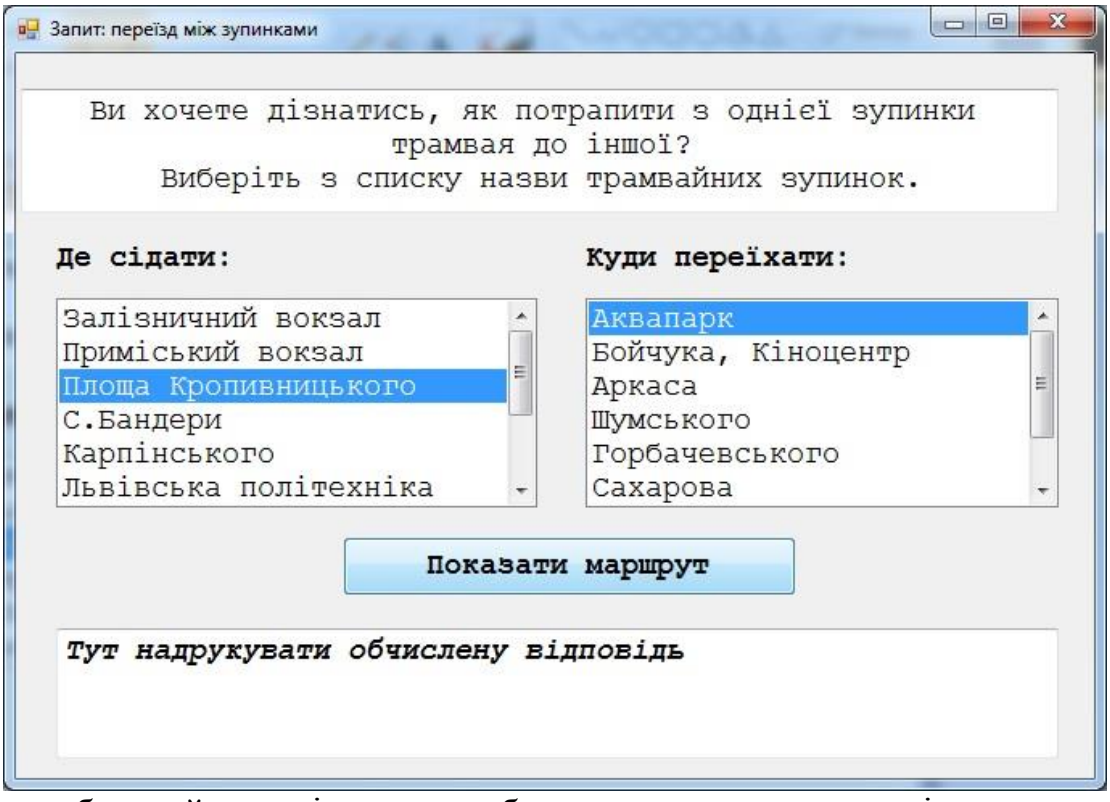
3-4.6.
запит) Який трамвай має маршрут через зупинки "Оперний театр",
"Варшавська"?
відповідь) Зупинки "Варшавська" немає в переліку трамвайних маршрутів.

3-4.7.
запит) Де можна купити квашеного огірка?
варіанти відповідей)
а) Ваше запитання не відповідає правилам запиту.
б) ☹ ???
в) Не знаємо! Закушуйте чим маєте! Радимо вести здоровий спосіб життя!

Побудувати повний перелік типів запитів, які будуть реалізовані в проєкті – самостійно. Визначити формати запитів і формати відповідей.

Деякі пояснення. 1)Формати і тексти запитів повинні мати інтерфейс дружній для будь-якого користувача, не можна будувати запити в термінах програмних конструкцій автора системи. 2)Добре організована система повинна вміти давати відповіді на будь-які запити, навіть якщо запити не за темою; користувач може подати запит помилково або для власної цікавості.

4.1.Подібні системи працюють в діалоговому режимі з використанням технології віконних елементів керування. Наприклад, для запиту виду 3-4.1 можна побудувати таке вікно:



Отже, треба прийняти рішення, як будувати це завдання в цілому – як віконну програму чи консольну програму. Оскільки наш курс орієнтований на мову Python,

то простішим варіантом буде консольна програма в режимі імітації віконних операцій. Якщо робити віконну програму на "чистому" Python, тоді треба використати модуль tkinter (*Interface to Tcl/Tk for graphical user interfaces*) бібліотеки модулів Python або інший модуль графічних операцій.

4.2.Режим імітації віконних операцій можна реалізувати, склавши окремі функції для кожного активного елемента умовного вікна. Наприклад, для елемента "Де сідати:" можна передбачити функцію

```
def WhereToSit(selectionlist, selecteditem):
  # деякі операції
  return selecteditemOrcode  # повертати можна код зупинки
```

Другий параметр `selecteditem` можна використати для імітації "вибору" назви зупинки, тобто це є тестовий варіант виконання функції.

Аналогічно можна будувати функцію для елемента "Куди переїхати:":

```
def WhereToMove(selectionlist, selecteditem):
```

А для кнопки "Показати маршрут" можна передбачити функцію

```
def ShowRouteFromTo(from, to):
  # параметри є назвами зупинок або їх кодами
  # обчислення відповіді про маршрут
  #   – головна обчислювальна частина цілого завдання
  return повний-текст-форматованої-відповіді
```

Цей проєкт є великий щодо підготовки даних для інформаційно-довідкової системи і формування програмних структур даних. [Але це роблять один раз.] Самі функції системи, як відповіді на потенційні запити, не виглядають складними. Творча робота може полягати у визначенні типів запитів і форматів надсилання відповідей в формі, яка має дружній інтерфейс, можливо, з варіаціями. Варіації – це відповідь на однакове запитання в різній формі, щоб занадто не набридати користувачам системи.

<u>Пропозиція</u>. За домовленістю можна скласти міні-команду в складі двох або трьох студентів. Студенти мають самостійно поділити між собою обсяг цілої роботи, але викладачу повідомити про склад міні-команди і розподіл обов'язків. Така пропозиція заохочується додатковими балами до оцінки, бо дає змогу отримати навики командної роботи. Всі члени міні-команди отримають однакову оцінку, як це прийнято в системі "замовник-виконавець".

Результати проєкту надіслати в такому вигляді:

- файл п.1, якщо такий є;
- повний текст файла програмного коду Python, оформлений одним файлом; в програмному коді мають бути пояснення у вигляді коментарів, про маршрути, списки зупинок, інші обчислені дані;
- перелік форматів запитів і відповідей;

- файл інструкції користувачу системи про те, як складати запит; реально цей файл не буде застосований для тестування системи, але на оцінку проєкту впливати буде;
- приклад сценарію запитів-відповідей і відповідний протокол роботи програми.