Для виконання даної лабораторної роботи були використані наступні списки з ТОП-ами паролів:

- ТОП-100: https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/10-million-password-list-top-100.txt
- ТОП-100000: https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/10-million-password-list-top-100000.txt

Генерація паролів відбувається у класі `PasswordsGenerator.cs` . У конструктор класу передаються такі додаткові параметри:

- `top100Persentage` (за замовчуванням 5) – відсоток паролів, які треба генерувати зі списку ТОП-100;
- `top100000Persentage` (за замовч. 70) – відсоток паролів, які треба генерувати зі списку ТОП-100000;
- `randomPasswordsPersentage` (за замовч. 5) – відсоток паролів, які необхідно генерувати рандомно.

Паролі генеруються за допомогою методу `GeneratePasswords` наступними чином:

- `top100Persentage` відсоток паролів заповнюється зі списку ТОП-100;
- `top100000Persentage` відсоток паролів заповнюється зі списку ТОП-100000;
- `randomPasswordsPersentage` відсоток паролів генерується рандомно за допомогою методу `CreateRandomPassword` , який створює пароль довжини `length` (переданої у вхідні параметри) із рядку допустимих символів `source = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890!:$_"` ;
- решта (100% - `top100Persentage` - `top100000Persentage` - `randomPasswordsPersentage`) відсоток паролів генерується на основі списку ТОП-100000 за допомогою правил, описаних у регіоні `#region Rules` та поміщених у список `passwordGeneratingRules` :
    1) `AppendNumbers` – додає на початок паролю 1-5 рандомних цифр;
    2) `PrependNumbers` – додає в кінець паролю 1-5 рандомних цифр;
    3) `Transliterate` – транслітерує латиницю в кирилицю;
    4) `ReplaceLetters` – замінює літери на схожі за виглядом цифри/символи;
    5) `ReplaceNumbers` – замінює цифри на схожі за виглядом літери;
    6) `ChangeCase` – замінює регістр літер у паролі таким чином, щоб верхній чергувався з нижнім (наприклад, “PaSsWoRd” або “pAsSwOrD”).

Кількість паролів `passwordsCount` , які потрібно згенерувати, передається у вхідні параметри методу `GeneratePasswords` .

Паролі хешуються за допомогою 3-ох алгоритмів:

1. MD5 (реалізація у класі `MD5Hasher`);
2. SHA1 (`SHA1Hasher`);
3. Argon2id (`Argon2idHasher`).

Вищезазначені класи наслідуються від базового `BaseHasher` , що містить у собі спільний для всіх алгоритмів метод для створення «солі» `CreateSalt` , де «сіль» створюється за допомогою `System.Security.Cryptography.RNGCryptoServiceProvider.GetBytes` .

Програма створює 3 набори паролів по 100_000 записів у кожному, потім хешує ці набори наведеними алгоритмами хешування та записує результати у 3 csv файли:

1. https://github.com/KasprukNastia/security/blob/master/Lab4/Lab4/Data/md5_hashed.csv – паролі захешовані алгоритмом MD5.
2. https://github.com/KasprukNastia/security/blob/master/Lab4/Lab4/Data/sha1_hashed.csv - SHA1.
3. https://github.com/KasprukNastia/security/blob/master/Lab4/Lab4/Data/argon2id_hashed.csv – Argon2id.