***Теоретичні засади***

Основою для виконання лаборатної роботи було математичне правило, що якщо ми маємо 2 повідомлення А та В, і їх хеш

*E(A) = A xor C*

*E(B) = B xor C*

, то

*E(A) xor E(B) = (A xor C) xor (B xor C) = A xor B xor C xor C = A xor B.*

Виходячи з цього правила, ми можемо підставляти часто вживані слова замість D в наступному виразі

*A xor B xor D*

, і якщо, наприклад, повідомлення А міститиме всередині слово D, то в результаті цього виразу ми отримаємо відповідну частину повідомлення В.

Звісно ж, це працює тільки якщо дотримуватися довжини D (обрізати всі байти довші за цю довжину) в виразі *A xor B xor D* та зміщувати *A xor B,* адже ми не знаємо, в якій саме позиції стоїть це слово. Функція, що це робить наведена на рисунку 1:

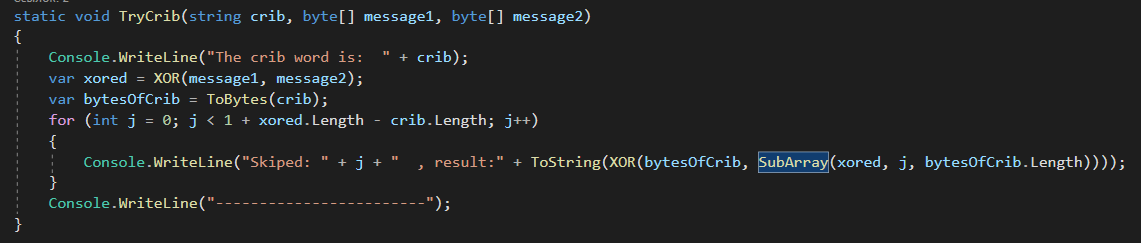


Рисунок 1 – функція, що застосовує часто вживане слово (кріб) для відгадування

***Демонстрація***

Програма допомагає відгадати рядки, що зашифровані одним ключем із тексту

ad924af7a9cdaf3a1bb0c3fe1a20a3f367d82b0f05f8e75643ba688ea2ce8ec88f4762fbe93b50bf5138c7b699

a59a0eaeb4d1fc325ab797b31425e6bc66d36e5b18efe8060cb32edeaad68180db4979ede43856a24c7d

a59a0eaeaad7fc3c56fe82fd1f6bb5a769c43a0f0cfae74f0df56fdae3db8d9d840875ecae2557bf563fcea2

a59a0eaea8ddf93c08fe81e11e2ab2bb6d962f0f1af2f44243b46cc1b6d6c291995d65a9a5234aa204

ad924af7a9cdaf3a1bb0c3f51439a5b628cf215a1fbdee4302a77a8ea2cc86c8984d65ffac6c58bf5b71dab8841136

b09b4afda3caf93c5aa78ce6096bb2a67ad86e4302f3e10602b37acbb1829680935137e8bb2919b6503fccfdca5461

a59a0eaeb5d7af3115b287b31425e6a460d3200f19f5e35406f567dde3cc8d9c9e4179eee92557f1463edc

a18c09ebb6ccaf2d12bbc3c41227aaf37fde274c05bdf5471aa62edaac82968093452da9eb0456bd5b71c6bfcb56

ad924af7a9cdaf3a1bb0c3e71a27adf37fdf3a474dfef44914b17d8ea2cc86c89d4d72f9e93556a44d71dfb8980034b3cea5c4d4

ab864af9a7d4e4790db797fb5b00afbd6fc5acaff9f3e95443b961dda6829680930874e6a42156bf1f25c6a4891c6d

ad924ae0a3d1fb311facc3f5142eb5f366d93c0f01f2f04f0db22ec8b1cb8786925b37eaa82219b94a23ddf1931b34fa

ad924aefaad4af341fb0c3f0143ea8a728c1275b05bdff4916f92eccb6d6c286994672a9bd2356f15224cab9d1

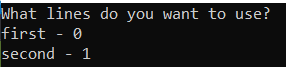
ad924af7a9cdaf3a1bb0c3f51227aaf37cde2b0f18f3e04911b267d8aacc85c89b4179fcbd29

b39d1ee6e6cbe6210ea7c3e01e28a9bd6cc5690f1af2f4520bf561c8e3c68b9b824979eaac6c4ba4517d89f1ca

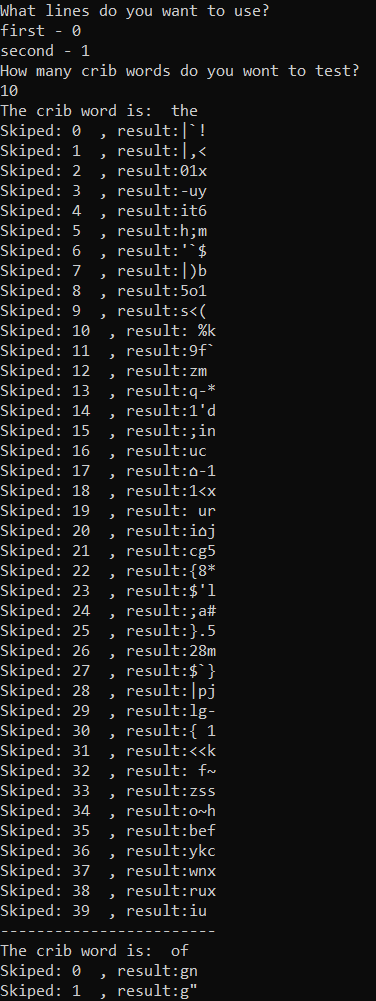
bd9b1ffcb598e62a5aaa8bf65b0ea7a17cde6e4e03f9a64315b07cd7b7ca8b86910863e1a8381ea21f38c7f183006df6c2a5

a59a0e6c462cf83113bd8bb31238e6be67c42bcded09ff4916f262c2e3c087c897085ae8a76019bc4671dabe8455

Спочатку вона питає в користувача які саме 2 рядки ми використовуватимемо:



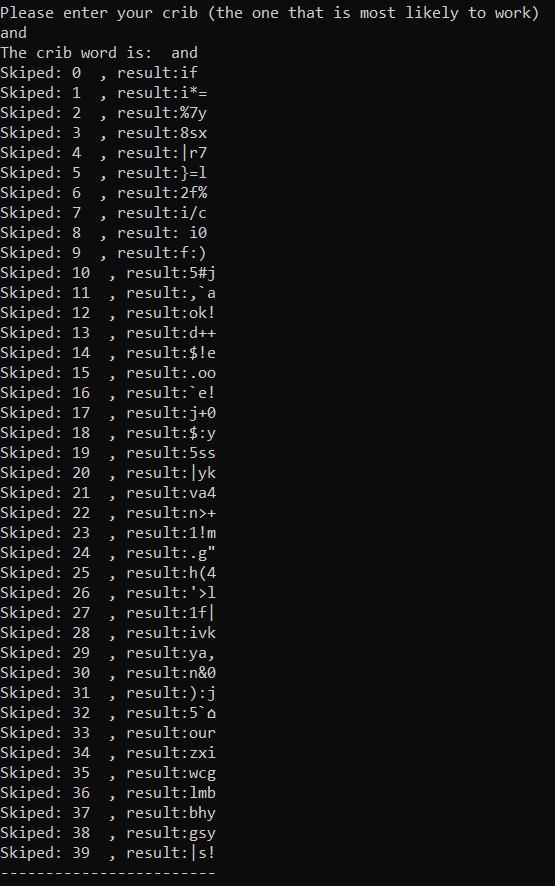
Далі програма пропонує використати найчастіші англійські слова і подивитись чи не схожий результат на англійський текст. Користувач має задати як багато часто використовуваних слів ми беремо із словника



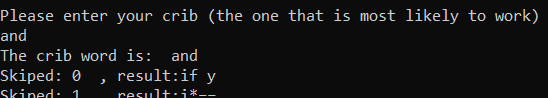
Далі користувач має подивитись на пропоновані варіанти і вибрати найбільш підходящий на його думку. Наприклад, нехай користувач вибере and оскільки «if » схоже на розшифровану фразу.



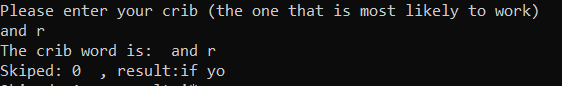
Тепер програма виводить результат тільки для введеного слова “and”



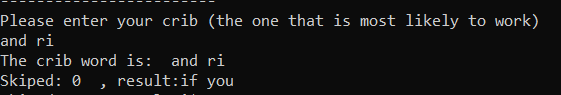
Логічним є, що далі має бути пробіл «and ». Спробуємо цю комбінацію:



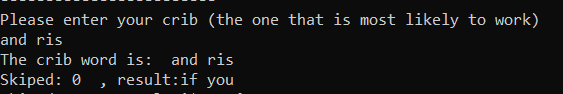
Припумо, що y це початок слова you. Для цього спробуємо підібрати літеру в crib, що дає літеру о в результаті



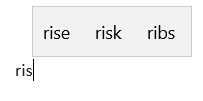
Далі спробуємо підібрати літеру в crib, що дає літеру u в результаті

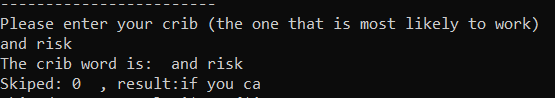


Далі тут слід було б поставити пробіл, тому підбираємо пробіл



Використаємо автодоповнення щоб здогадатись про наступну літеру





Бачимо, що “and risk ” видає непоганий результат.

Таким чином, ми можемо відновити покроково вихідний текст, хоч це і займе немало часу.