Problem 1. (3 parts, 120 points total.) For $k \in \mathbb{N}$, let $F : \mathcal{K} \times \{0,1\}^* \rightarrow \{0,1\}^k$ be a pseudorandom function and $\mathcal{SE} = (\mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme with key-length $k$ and message-space $\{0,1\}^*$. Let $\mathcal{DB} = (w_i, \mathcal{ID}_i)_{i=1,...,n}$ be a database (aka. inverted index) of $n$ keywords. (We leave the space of keywords and document identifiers here arbitrary and implicit.) Define a searchable symmetric encryption scheme $\mathcal{SSE}^* = (\text{KeyGen}, \text{Enc}, \text{TrapGen}, \text{Search})$ as follows:

**Algorithm KeyGen:**

$K \leftarrow \$ \mathcal{K}$

Return $K$

**Algorithm Enc($K, \mathcal{DB}$):**

Initialize history-independent dictionary $\text{dict}$

For $i = 1$ to $n$ do:

$K_{E,i} \leftarrow F(K, w_i||1)$

$K_{I,i,1} \leftarrow F(K, w_i||2)$

$K_{I,i,2} \leftarrow F(K, w_i||3)$

$r_i \leftarrow \{0, \ldots , |\mathcal{DB}(w_i)| - 1\}$

For $j = 1$ to $r_i$ do:

$c_{i,1}[j] \leftarrow \$ \mathcal{E}(K_{E,i}, \mathcal{DB}(w_i)[j])$

For $j = r_i + 1$ to $|\mathcal{DB}(w_i)|$ do:

$c_{i,2}[j] \leftarrow \$ \mathcal{E}(K_{E,i}, \mathcal{DB}(w_i)[j])$

Insert $(K_{I,i,1}, c_{i,1})$ and $(K_{I,i,2}, c_{i,2})$ into $\text{dict}$

Return $\text{dict}$ as encrypted database $\text{EDB}$

**Algorithm TrapGen($K, w$):**

: *

**Algorithm Search($T_w, \text{EDB}$):**

: *

Recall above that $\mathcal{DB}(w)$ denotes the postings list for a keyword $w$, i.e., $\mathcal{DB}(w_i) = \mathcal{ID}_i$ for all $i \in \{1, \ldots , n\}$, which the encryption algorithm accesses as an array.

**Part A:** (15 points.) Fill in the appropriate pseudocode for the TrapGen and Search algorithms.

**Part B:** (70 points.) Recall that non-adaptive SIM-security of a searchable symmetric encryption scheme $\mathcal{SSE}$ \footnote{In the subsequent experiments we use the same names for the constituent algorithms of $\mathcal{SSE}$ as for $\mathcal{SSE}^*$ above, with the understanding that $\mathcal{SSE}$ is arbitrary.} wrt. a leakage profile $\mathcal{L}$ is defined via the following experiments associated to an adversary $A = (A_1, A_2)$ and simulator $S$: 
Experiment $\text{Exp}_{\text{SSE}}^\text{sim-1}(A)$

$K \leftarrow \text{KeyGen}$

$(DB, w, state) \leftarrow s A_1$

$EDB \leftarrow \text{Enc}(K, DB)$

For $j = 1$ to $|w|$ do:

\begin{align*}
T[j] & \leftarrow \text{TrapGen}(K, w[j]) \\
(f, y) & \leftarrow s A_2(EDB, T, state) \\
\text{If } f(DB) = y \text{ then return 1} \\
\text{Else return 0}
\end{align*}

Experiment $\text{Exp}_{\text{SSE}, S}^\text{sim-0}(A)$

$(DB, w, state) \leftarrow s A_1$

$(EDB, T) \leftarrow s S(L(DB, w), state)$

$(f, y) \leftarrow s A_2(EDB, T, state)$

If $f(DB) = y$ then return 1

Else return 0

We say that SSE is non-adaptive SIM-secure wrt. $L$ if for every efficient $A$ there is an efficient $S$ such that

$$\text{Adv}_{\text{SSE}, S}^\text{sim}(A) = \Pr[\text{Exp}_{\text{SSE}}^\text{sim-1}(A) \text{ outputs 1}] - \Pr[\text{Exp}_{\text{SSE}, S}^\text{sim-0}(A) \text{ outputs 1}]$$

is small.

Give a minimal (i.e., containing as little information about DB as possible) leakage profile $L$ such that SSE* can be proven non-adaptive SIM-secure wrt. $L$. Here $L$ is a (possibly randomized) algorithm that takes the unencrypted database DB and the adversary’s queries $w$ and returns some corresponding leakage. You should fill in the pseudocode for the algorithm below:

Algorithm $L(DB, w)$:

```
```

Note that you do not need to provide the corresponding proof of security. However, if you do so you will get 50 extra credit points.

Part C: (35 points.) Compare SSE* to the two SSE schemes we saw in class that use a history-independent dictionary. Recall that the first scheme has perfect locality (but is not parallelizable) and encrypted database size $O(|DB|^2)$, including the padding for each postings list. The second scheme is fully parallelizable (but is completely non-local) and has encrypted database size $O(|DB|)$. Your comparison should be in terms of efficiency (namely locality, parallelizability, and encrypted database size) and security (namely how much information is contained in the leakage profile). Do you think SSE* may be superior in any practical setting? Why or why not?