CSE 208 Online on Hashing

Overview:

Implement a MemoryManager to allocate fixed-size memory blocks to processes using **open addressing with quadratic probing**. The memory is stored in a hash table with dynamic resizing to maintain performance where each slot can be in one of three states: **EMPTY**, **OCCUPIED**, or **DELETED**.

Operations:

(1) ALLOCATE process_id (Command: ALLOCATE <process_id>)

- Use **quadratic probing**:
  index = (hash(process_id) + j^2) % table_size, j = 0, 1, 2, …
  Use hash(k)=k for simplicity.
- Insert into the first EMPTY or DELETED slot. Mark that slot as OCCUPIED.
- Count probes: if **probes > P_MAX (taken as input)**, trigger **scale-up rehash (described below)**.

(2) FREE process_id (Command: FREE <process_id>)

- Locate using the same probing.
- Mark the slot as DELETED (not EMPTY).
- After every **5 deletions**, check load factor:
  - If **load factor < 0.2**, trigger **scale-down rehash (described below)**.
  - Don't shrink below the initial size N.

Rehashing Rules:

(1) Scale-Up

- **Trigger**: Probe count exceeds P_MAX.
- **Action**: Resize to next prime ≥ 2 * current size. Then, rehash all OCCUPIED processes from the old table into the new, larger table using new size. DELETED blocks are not carried over; they effectively become EMPTY in the new table.

(2) Scale-Down

- **Trigger**: After every **5** deletions, check load factor. If load factor **< 0.2**, scale down.
- **Action**: Resize to closest smaller prime <= current size / 2.
  Skip if new size < N.

- Then, rehash all OCCUPIED processes from the old table into the new, smaller one. DELETED blocks are not carried over; they effectively become EMPTY in the new table.

Input Format:

The input consists of a sequence of operations, one per line. The first line contains three integers: **N P_MAX Q**

- **N: Initial size of the memory table (must be a prime number)**
- **P_MAX: Maximum allowed probe count before triggering a rehash**
- **Q: Number of operations.**

Each subsequent line contains one operation, in one of the following formats:

- **ALLOCATE <process_id>**
- **FREE <process_id>**

Output Format:

On ALLOCATE <process_id>:

Print ALLOCATE <process_id>: Inserted at index <index> (probes: <count>)

- **<index> is the position where the process was inserted.**
- **<count> is the number of probes it took to find the slot.**

If rehashing is triggered due to probe count exceeding P_MAX, print the following before and after rehashing:

--- Rehash Triggered: BEFORE ---
Table Size: <old_size>
Active Processes: <occupied_count>
Load Factor: <load_factor>
Max Probe Sequence: <max_probe_count>
--- Rehash Triggered: AFTER ---
Table Size: <new_size>
Active Processes: <occupied_count>
Load Factor: <load_factor>
Max Probe Sequence: <max_probe_count>

On FREE <process_id>:

Print FREE <process_id>: Freed from index <index>

If rehashing is triggered after 5 deletions and load factor drops below 0.2, also print the same Rehash Triggered report as above.

Hints:

- **Design a Slot Structure :** Store process_id and state for each slot.
- **Initialize the Table:** Initialize the slots with EMPTY status
- **ALLOCATE Function:**

    (i) Use quadratic probing to find an EMPTY or DELETED slot.

    (ii) Insert and mark OCCUPIED.

    (iii) Count probes; if `probes > P_MAX`, trigger **scale-up rehash**.

- **FREE Function:**

    (i) Use quadratic probing to find the process.

    (ii) Mark as DELETED if found.

    (iii) Track deletions; after every 5, check load factor and **scale-down if needed**.

- **Rehashing:**

    (i) Create new table of target size.
    (ii) Re-insert only OCCUPIED items to the new table by calculating new hash values based on the new size(drop DELETED).
    (iii) Update all metadata and reset deletion counter.

- **Statistics Tracking**

    Keep counters for:

    - OCCUPIED slots
    - Max probes seen so far
    - Total deletions