Extendible Hashing: Lab Assignment

We are assuming that you have coded the extendible hashing (EH) scheme as described in the Rahu Ramakrishnan's book.

Now, you are expected to customize the directory doubling in the extendible hashing. Directory doubling is the most expensive part of the EH. We want to delay it as much as possible. We will have the following criteria for directory doubling:

We will double the directory only when at least M percentage of existing buckets have at least one overflow bucket attached to them.

Value of M is a parameter to your program through input file.

Here is how you are expected to handle the insertion operation:

Case 1: The destination bucket has space for the key.

Solution: Just insert the key.

Case 2: The destination bucket is full. The local depth is less than the global depth.

Solution: Split the bucket. Try to insert the key.

Case 3: The destination bucket is full. The local depth is equal to the global depth.

Solution:

- Create an overflow bucket. Attach it to the current full bucket.
- Insert the key into the overflow bucket.
- Now check the total number of buckets with overflow (say P). Let the total number of buckets in the EH be Q.
- If P*100/Q is less than M then no need to double the directory. Otherwise double the directory and split every bucket that had overflow bucket attached to it.

You are not expected to handle deletions or search for this assignment.

Assume that there are no duplicates in the input.

Example input 1:

First two lines describe the initialization parameters for your extendible hash table.

Line 1: Global depth

Line 2: Bucket capacity

Line 3: Value of M as percentage

Remaining lines of the input file describe the operations to be performed on the hash table.

2: Insert new value

5: Display status of the hash table (Just mention the global depth and the number of buckets)

6: Quit

For example:

- Line 4 asks you to insert value 1 in the hash table
- Line 6 asks you to display the status of the hash table.
- Line 22 asks you to quit the program.

Directory can grow only up to the Global Depth of 22.

Example output 1:

Line 1: After inserting values 1 and 2, the hash table has global depth of 1 and 2 buckets.

(Bucket B0 for directory slot 0 and B1 for directory slot 1.)

Line 2: Value 3 is inserted in bucket B1. B1 was already full. So, we attach an overflow bucket to it.

P=1, Q= 2. P*100/Q = 50. Value of M is 60. We will not double the directory. We still have global depth of 1 and 2 buckets.

Line 3: Value 4 is inserted in bucket B0. Now, we will have two buckets with overflow.

P=2, Q=2. P*100/Q = 100. Value of M is 60. Now, we have to double the directory and split every bucket that has overflow attached to it.

(We have the following mapping between directory slots and buckets 00: B0, 01: B1, 10: B2, and 11:B3).

After splitting, each bucket will have one key (B0: 4, B1: 1, B2: 2, and B3: 3)

Line 4: Insertion of values 5 and 6 cause overflow in corresponding buckets (B1 and B2). However, we still have not crossed the 60 percent threshold for directory doubling.

P=2, Q=4, P*100/Q = 50

Line 5: Insertion of value 7 cause overflow in bucket B3.

P=3, Q=4, P*100/Q = 75.

Now, we have to double the directory. Initially, we will copy the existing bucket pointers

000: B0

001: B1

010: B2

011: B3

100: B0

101: B1

110: B2

111: B3

However, buckets B1, B2, and B3 need split. We will create corresponding three new buckets and split the keys.

000: B0 (keys: 4)

001: B1 (keys: 1)

010: B2 (keys: 2)

011: B3 (keys: 2)

100: B0

101: B4 (keys: 5)

110: B5 (keys: 6)

111: B6 (keys: 7)

Line 6: Insertion of value 8 causes overflow in bucket B0. However, local depth for B0 is less then global depth. We will simply split the bucket.

000: B0 (keys: 8)

001: B1 (keys: 1)

010: B2 (keys: 2)

011: B3 (keys: 2)

100: B7: (keys: 4)

101: B4 (keys: 5)

110: B5 (keys: 6)

111: B6 (keys: 7)

Line 7:

Insertion of keys 9, 10, and 11 causes overflow in buckets B1, B2, and B3 respectively.

P=3, Q=8, P*100/Q = 37.5.

We will not double the directory at this stage.

Note on code submission:

Make sure that all your code fits into a single file <roll number>EH.cpp

How your code should compile?

```
g++  <roll number>EH.cpp
```

How your code should run?

```
a.out < input_file_name
```

Your code should write the output on the stdout