CS293: Data Structures and Algorithms Lab

Lab 3 - Limited Space

This problem extends on the implementation of `Dictionary` in Lab 3.

BEFORE YOU START: Copy your implementation of `dictionary.cpp` to the current directory. Make sure to modify the datatype of `key` from `char*` to `std::string` in `dictionary.cpp`.

Problem Description

In Lab 3 we have already implemented a `Dictionary` which stores struct of form `<key, value>`.

For this problem, we have an additional constraint on our dictionary. The total number of distinct keys stored at one point cannot be more than a given `capacity`.

To ensure this we might have to evict some entries of the dictionary before we can add new ones. For this problem, we will focus on 2 eviction policies -

1. **First-In-First-Out (FIFO)**: Elements added first to the dictionary should be removed earlier. Modifying the value of an element should not change the eviction order of its earlier instance.
2. **Least Recently Used (LRU)**: Elements with the last access at the earliest time should be removed first. Modifying the value of an element counts as an access and should make it least likely to be removed.

LimitedDictionary Class

We will implement the class `LimitedDictionary` by inherting the methods of `Dictionary`. For this problem you will only need to modify the file `limitedDictionary.cpp`.

```
enum Policy {FIFO, LRU};

struct ListEntry{
    std::string key;
    ListEntry* next;
    ListEntry* prev;
};

class LimitedDictionary : public Dictionary {
  private:
    int size;
    int capacity;

    Policy policy;

    // FIFO
    std::queue<std::string> q;

    void evict_fifo();
    void insert_fifo(struct Entry e);

    // LRU
    ListEntry* head;
    ListEntry* tail;

    void init_lru();
    void evict_lru();
    void insert_lru(struct Entry e);

  public:
    LimitedDictionary(int capacity, Policy policy);

    bool put(struct Entry e);
    bool remove(std::string key);
};
```

Figure 1: limitedDictionary.h

Structure of the class is already defined in `limitedDictionary.h`. You need to implement these methods in `limitedDictionary.cpp`.

The `size` attribute stores the current size of dictionary while `capacity` stores maximum number of unique keys. `Policy` stores the eviction policy followed for this dictionary.

FIFO will be implemented using a queue and **LRU** using a doubly linked list.

You have to code the constructor for this class and overload the `put` and `remove` functions for both the policies.

Read the `MakeFile` to see various commands given to compile and test your implementation for different policies. You can also go through the file `limitedDictTest.cpp` to understand how the code is tested.