CS 321 Data Structures (Spring 2017)

Programming Assignment #1, Due on Friday Jan 19th 2018 (11:59PM)

Introduction:
This programming assignment asks you to design a **cache** implementation using a linked list data structure. That is, write a `Cache` class having at least the following public methods – constructor, getObject, addObject, removeObject, clearCache and some others. The data to be stored in cache should be generic objects. Also, write a test program to test your cache implementation.

Description:
A cache is a storage in memory. If a data item has a copy in the cache, an application can read this data item from cache directly. The usage of the cache is as follows. Whenever an application requires a data item, it searches the cache first. If it is a cache hit, then the cache returns the data item to the application and the data item will be move to the first position in the cache (we call it the Most Recently Used MRU scheme). On the other hand, if it is a cache miss, then the application needs to read the data item from disk and then the data item is added to the first position of the cache. Note that if the cache is full, the last entry (oldest one) in the cache is removed before a new entry can be added.

Similarly, whenever an application writes a data item to disk, the system will perform the same write operation to the cache copy of the data item (if any) and then move it to the first position in cache. Note that the `write` operation is equivalent to a `remove` operation followed by an `add` operation.

One-level Cache:
A single-level cache and it works as described above.

Two-level Cache:
A 2nd-level cache sits behind the 1st-level cache. Usually, the 2nd-level cache is much bigger than the 1st-level cache. Assume the 2nd-level cache contains all data in the 1st level cache, which is called `multilevel inclusion property`. Two-level cache works as follows:

1) If 1st-level cache hit: Both cache have the hit data item. Move the hit data item to the top on both cache.

2) If 1st-level cache miss and 2nd-level cache hit: The data item is not in 1st-level cache but is in 2nd-level cache. Move the data item to the top of 2nd-level cache and add the item to the top of 1st-level cache.

3) If 1st-level cache miss and 2nd-level cache miss: Retrieve the data item from disk and add the item to the top of both cache.

Hit Ratio:
Some terms used to define hit ratio are:
HR: (global) cache hit ratio
HR_1: 1st-level cache hit ratio
HR_2: 2nd-level cache hit ratio
NH: total number of cache hits
NH_1: number of 1st-level cache hits

NH_2: number of 2nd-level cache hits
NR: total references to cache
NR_1: number of references to 1st-level cache
NR_2: number of references to 2nd-level cache (= number of 1st-level cache misses)

- One-level cache: $HR = \frac{NH}{NR}$
- Two-level cache: $HR1 = \frac{NH_1}{NR_1}$ $\quad HR_2 = \frac{NH_2}{NR_2}$ $\quad HR = \frac{NH_1+NH_2}{NR_1}$

What you need to do:

1 Write a `Cache` class.

2 Download the Encyclopedia.txt file from the assignment page. Please do not include this file in your project submissions.

3. Write a test program. It's usage should be

   `java Test 1 <cache size> <input textfile name>` or
   `java Test 2 <1st-level cache size> <2nd-level cache size> <input textfile name>`

   The cache size(s) and the text file should be input as command line arguments. Your program should create a cache (option 1) or two cache (option 2) with the specified size(s) and read in the input text file word by word. For each word, search the cache(s) to see whether there is a cache hit and update the cache accordingly. I will use the file `Encyclopedia.txt` and a small text file to test your program.

5. Your program should output the cache hit ratio(s) after reading all words from the input text file. You can find the sample outputs, `result1k2k` and `result1k5k`, on the assignment page.

Submission:

Submit your assignment on Blackboard through the submit page for the assignment. You must include a README file describing how your program works and explaining any issues or design decisions you made in implementing your solution.