CMPSC-132: Programming and Computation II
Spring 2020

Homework 4

Due Date: 04/25/2020, 11:59PM EST
100 pts

Read the instructions carefully before starting the assignment. Make sure your code follows the stated guidelines to ensure full credit for your work.

Instructions:

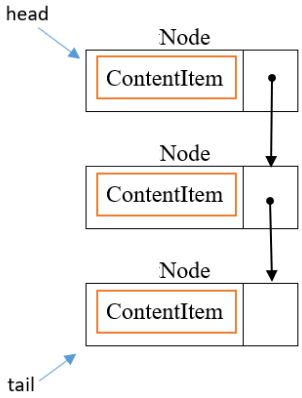
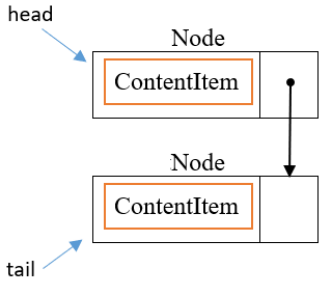
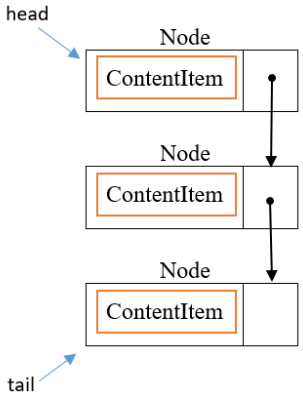
- The work in this assignment must be completed alone and must be your own. . If you have questions about the assignment, contact the instructor or Teaching/Learning Assistants instead of asking for a solution online
- **Download the starter code file from the HW4 Assignment on Canvas. Do not change the function names or given started code on your script. Also, please do not change the string formats given! These will be useful for displaying the content in a convenient and consistent way.***
- You are responsible for debugging and testing your code with enough data, you can share testing code on Piazza.
- A doctest is provided as an example of code functionality. You are responsible for debugging and testing your code with enough data.

Goal:

- ✓ In the Module 5 and 6 video lectures, we discussed the linked list and hash table data structure. This homework assignment will combine both concepts to create a (heavily) modified Cache class. In a computer, a cache stores data so that any future accesses of that data can be completed very quickly. A practical example that you've probably noticed is your browser cache. This stores a lot of frequently used, useful information in your browser so it's ready to access whenever you need it (frequent searches, passwords, usernames, etc.). If you've ever cleared your cache, you'd notice that all of this information is gone and needs to be repopulated. The implementation of the cache for this project will be very simplified and would ignore some of the features of a cache you might find online – so make sure to read this assignment document carefully.
- ✓ Before jumping into some of the specifics, it's important to understand how a cache is organized. Most caches on a computer are split into different levels. For this assignment, we will have **three** levels of caches, **L1**, **L2,** and **L3**. Usually, in a cache, if the computer is looking for data and it is unable to find it in one level, it'll check another level to find it. A computer often keeps the most frequently used information in the **L1** cache because it's the fastest and first level it will check. We won't be implementing this aspect of a cache for this assignment, but it's useful information to understand how a cache works in general. **The hierarchy of caches as different levels (L1, L2, L3) will be implemented as *hash table* in this assignment. In this implementation, you will create content and put it into either the L1, L2, or L3 cache based on one of the content's attributes (more on this later).**
- ✓ At each level of the cache, you'll have to actually implement a data structure to store the data. **For this assignment, the cache at each level of the hierarchy will be implemented as a *linked list.*** Note that this assignment is almost 70% the code written during the Module 5 Hands On session, you only need to make a few adjustments for this structure

✓ Given this information, your cache can be visualized like this:

Cache Hash Table

0	1	2
Level L1	Level L2	Level L3
CacheList 	CacheList 	CacheList 

✓ Each level's cache will have a size of 200. It **cannot** exceed this size. This is initialized for you in the code.

✓ The contentItem class has been created for you as well, and objects of this class store the following information:

- content_id: int
- size: numeric
- header: string
- content: string

✓ These are some examples of ContentItem objects formatted using the __str__/__repr__ functions given in the code:

```
>>> content1 = ContentItem(1006, 170, "Content-Type: 2", "<html><button>'Click Me'</button></html>")
>>> content1
CONTENT ID: 1006 SIZE: 170 HEADER: Content-Type: 2 CONTENT: <html><button>'Click
Me'</button></html>

>>> content2 = ContentItem(1009, 10, "<push>", "0xA")
>>> content2
CONTENT ID: 1009 SIZE: 10 HEADER: <push> CONTENT: 0xA
```

✓ In summary, you will complete the given method in the Cache and CacheList classes. The Cache objects will take ContentItem objects and insert them into the appropriate cache (L1, L2, or L3). This directory of caches will be implemented using a hash table. To insert, evict, and find content within a cache you will need to implement it as a linked list..

Brief Description of required methods

class Cache

hashFunc(contentHeader):
Sums the ASCII values of each content header % size
Parameters:
contentHeader: *str*
Returns:
int between 0 and 2 (inclusive)

insert(content, evictionPolicy):
Adds a contentItem into the proper Linked List
Parameters:
content: *contentItem*
evictionPolicy: *str*
Returns:
str INSERTED + contentItem representation

retrieveContent(content):
Gets the content from the hash table
Parameters:
content: *contentItem*
Returns:
contentItem if content is in the hash table, *str* otherwise

updateContent(content):
Updates the contentItem in the hash table
Parameters:
content: *contentItem*
Returns:
str UPDATED + contentItem representation if content is in the hash table, *str* otherwise

class CacheList:

put(content, evictionPolicy):
Adds Nodes at the beginning of the list. If necessary, uses mru/lru to remove items before adding
Parameters:
content: *contentItem*
evictionPolicy: str
Returns:
contentItem inserted

find(cid):
Search for content in the list. If content is found, Node is moved at the beginning of the list
Parameters:
cid: *int*
Returns:
contentItem is cid is in the list, None otherwise

update(cid, content):

Updates the content in the list. If content is found and updated, Node is moved at the beginning of the list

Parameters:

cid: *int*

content: *contentItem*

Returns:

contentItem is cid is in the list, None otherwise

mruEvict():

Removes the first item of the list

Parameters:

None

Returns:

None

lruEvict():

Removes the last item of the list

Parameters:

None

Returns:

None

clear():

Removes all items from the list.

Parameters:

None

Returns:

None

The next section will cover specifics about the implementations of the hash table and linked list

The Cache Directory (Hash Table)

The class Cache() is the data structure you will use to store the three other caches (L1, L2, L3). It stores an array of 3 CacheList objects, which are the linked list implementations of the cache (which will be explained later).

hierarchy

0	1	2
Level L1	Level L2	Level L3
CacheList(200)	CacheList(200)	CacheList(200)

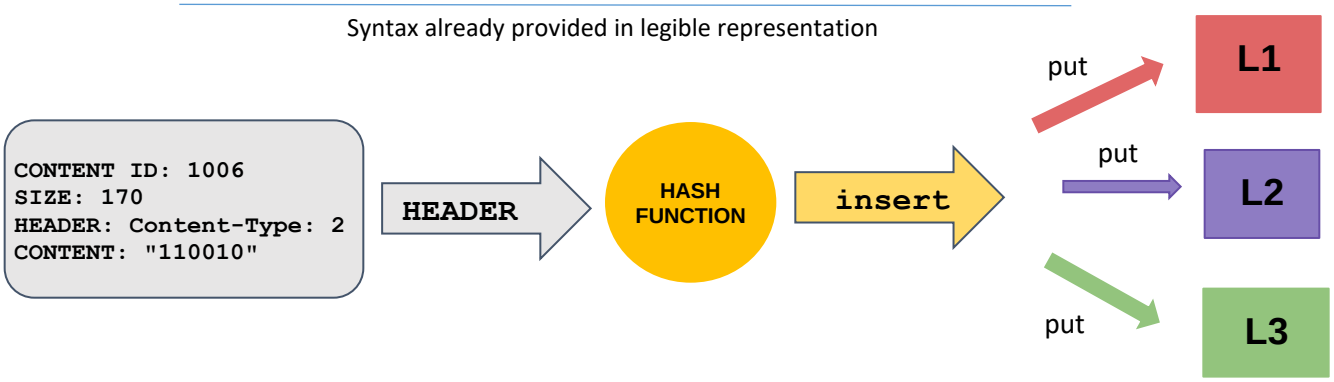
The initialization of CacheList objects has been already written for you (a Python list with 3 CacheList objects, size 200 each). Do not change this. There are 3 functions you must implement yourself:

- def hashFunc(self, contentHeader)
  - This function determines which cache level your content should go in. Create a hash function that sums the ASCII values of each content header, takes the modulus of that sum with the size of the Cache hash table, and accordingly assigns an index for that content – corresponding to the L1, L2, or L3 cache. So, for example, let's assume the header "Content-Type: 2" sums to an ASCII value of 1334. 1334 % 3 = 2. So, that content would go in the L3 cache. If the header is "<push>", the sum of its ASCII values is 570, 570%3=0, so that content would go in the L1 cache. You should notice a very obvious pattern in which content headers correspond to which caches.
  - The hash function should be used by your *insert* and *retrieveContent* functions.

- def insert(self, content, evictionPolicy)
  - Once a ContentItem object is created, call the cache directory's insert function to insert it into the **proper cache**. This function should call the linked list's implementation of the insert function (put) to actually insert into the cache itself. The eviction policy will be a string – either **'lru'** or **'mru'**. These eviction policies will be explained later. You should check to make sure there is not another content item with the same content ID already in the list. If there, is return a message as per the doctest formatting.
  - This function should return a message including the attributes of the content object inserted into the cache. This is shown in the doctests.
  - The hash function tells your content where to go! Don't overthink this part.

```
>>> cache = Cache()
>>> content1 = ContentItem(1006, 170, "Content-Type: 2", "110010")
>>> cache.insert(content1, 'mru')
'INSERTED: CONTENT ID: 1006 SIZE: 170 HEADER: Content-Type: 2 CONTENT: 110010'
```

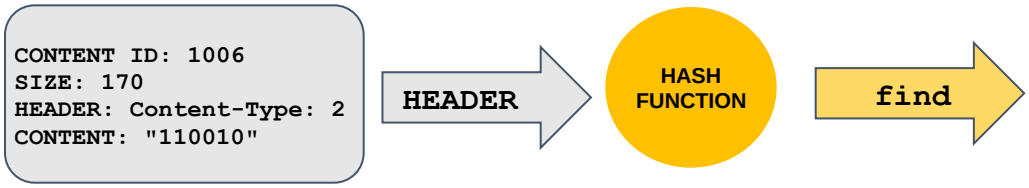
Syntax already provided in legible representation



- def retrieveContent(self, content)
    - This function should take a ContentItem object, determine which level it is in, and then return the object if it is in the cache at that level. **When the item is found, move it to the head of the list.** If not, it should return a message indicating that it was not found. This is known as a "cache miss". Accordingly, finding content in a cache is known as a "cache hit". The doctests show the format of the return statements.

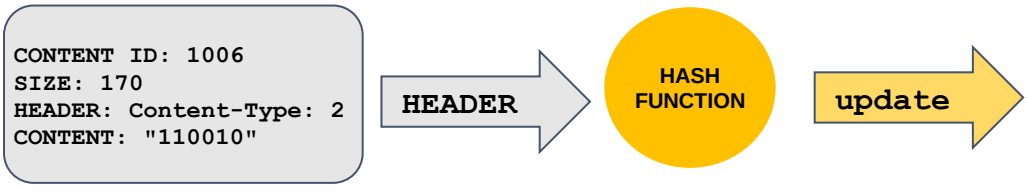
```
>>> cache = Cache()
>>> content1 = ContentItem(1006, 170, "Content-Type: 2", "110010")
>>> cache.insert(content1, 'mru')
'INSERTED: CONTENT ID: 1006 SIZE: 170 HEADER: Content-Type: 2 CONTENT: 110010'
>>> cache.retrieveContent(content1)
CONTENT ID: 1006 SIZE: 170 HEADER: Content-Type: 2 CONTENT: 110010
```

Syntax already provided in legible representation, just return the contentItem object



- def updateContent(self, content)
    - This function should take a content object, determine which level it is in. Find the content using its content ID and update the contentItem inside the Node object. You should replace the entire contentItem. If the content item is not in the cache, return a message indicating that it was not found (a cache miss). The doctests show the format of the return statements. **You do not need to worry about the size changing in the new content item! To keep the update function simple so that you don't have to worry about managing new content sizes and eviction, we will always replace content with content of the same size.**

```
>>> cache = Cache()
>>> content1 = ContentItem(1006, 170, "Content-Type: 2", "110010")
>>> cache.insert(content1, 'mru')
'INSERTED: CONTENT ID: 1006 SIZE: 170 HEADER: Content-Type: 2 CONTENT: 110010'
>>> content5 = ContentItem(1006, 170, "Content-Type: 2", "11111111111111")
>>> cache.updateContent(content5)
'UPDATED: CONTENT ID: 1006 SIZE: 170 HEADER: Content-Type: 2 CONTENT: 11111111111111'
```



The Cache List (Linked Lists):

As explained earlier, at each level within the cache directory, the actual cache is stored as a linked list object. You will use the CacheList() class to implement this linked list. Each cache has a maximum size of 200 **(Please note that this does NOT mean there can be a maximum of 200 items in the cache. It means that the sum of the sizes of all content objects in a cache cannot exceed 200)**. Do not change this value. There are 5 functions you must implement yourself.

- def put(self, content, evictionPolicy)
  - This function puts content into your cache. As explained earlier, caches should make information readily available to a user. Accordingly, you should always insert new content at the front of your list (head) so it is easy to obtain.

```
>>> cache = Cache()
>>> content1 = ContentItem(1006, 170, "Content-Type: 2", "110010")
>>> content2 = ContentItem(1004, 30, "Content-Type: 2", "11111")
>>> cache.insert(content1,'mru')   # Internally calls put()
>>> cache.insert(content2,'mru')   # Internally calls put()
>>> cache
…
L3 CACHE:
REMAINING SPACE:0
ITEMS:2
LIST:
[CONTENT ID: 1004 SIZE: 30 HEADER: Content-Type: 2 CONTENT: 110010]
[CONTENT ID: 1006 SIZE: 170 HEADER: Content-Type: 2 CONTENT: 110010]
```

0	1	2
CacheList(200)	CacheList(200)	CacheList(200) content2 ↓ content1

  - You should be checking to make sure that your content is not bigger than the maximum size, in that case it should not be inserted.

```
>>> content3 = ContentItem(1002, 715, "Content-Type: 2", "1101")
>>> cache.insert(content3,'mru')
'Insertion not allowed. Content size is too large.'
```

  - Similarly, if the content id is already in the list, insertion is not allowed, even if there is space in the list

```
>>> cache = Cache()
>>> content1 = ContentItem(1006, 70, "Content-Type: 2", "110010")
>>> content2 = ContentItem(1006, 30, "Content-Type: 2", "11111")
>>> cache.insert(content1,'mru')
>>> cache.insert(content2,'mru')
'Insertion of content item 1006 not allowed. Content already in cache.'
```

  - Accordingly, you should also make sure there is enough space remaining in the cache to insert your content. If there is not enough space, you need to *evict* content currently in the cache. You must keep removing content until there is enough space to insert the new content (so long as the new content is not bigger than the maximum size). There are two ways to do this: LRU or MRU.
    - LRU: Least Recently Used
      - Remove the content that was last added to the list (tail)
    - MRU: Most Recently Used
      - Remove the content that was most recently added to the list (head)

- These eviction policies will be specified when the hashtable's insert function is called. It will be passed as a string parameter 'lru' or 'mru'.

```
>>> cache = Cache()
>>> content1 = ContentItem(1006, 170, "Content-Type: 2", "110010")
>>> content2 = ContentItem(1004, 30, "Content-Type: 2", "11111")
>>> cache.insert(content1,'mru')   # Internally calls put()
>>> cache.insert(content2,'mru')   # Internally calls put()
>>> cache
…
L3 CACHE:
REMAINING SPACE:0
ITEMS:2
LIST:
[CONTENT ID: 1004 SIZE: 30 HEADER: Content-Type: 2 CONTENT: 110010]
[CONTENT ID: 1006 SIZE: 170 HEADER: Content-Type: 2 CONTENT: 110010]
```

0	1	2
CacheList(200)	CacheList(200)	CacheList(200) content2 ↓ content1

```
>>> content3 = ContentItem(1002, 15, "Content-Type: 2", "1101")
>>> cache.insert(content3,'lru')
'INSERTED: CONTENT ID: 1002 SIZE: 15 HEADER: Content-Type: 2 CONTENT: 1101'
>>> cache
…
L3 CACHE:
REMAINING SPACE:155
ITEMS:2
LIST:
[CONTENT ID: 1002 SIZE: 15 HEADER: Content-Type: 2 CONTENT: 1101]
[CONTENT ID: 1004 SIZE: 30 HEADER: Content-Type: 2 CONTENT: 110010]
```

0	1	2
CacheList(200)	CacheList(200)	CacheList(200) content3 ↓ content2

```
>>> content4 = ContentItem(4321, 200, "Content-Type: 2", "1000001")
>>> cache.insert(content4,'mru')
'INSERTED: CONTENT ID: 4321 SIZE: 200 HEADER: Content-Type: 2 CONTENT: 1000001'
>>> cache
…
L3 CACHE:
REMAINING SPACE:0
ITEMS:1
LIST:
[CONTENT ID: 4321 SIZE: 200 HEADER: Content-Type: 2 CONTENT: 1000001]
```

0	1	2
CacheList(200)	CacheList(200)	CacheList(200) content4

- def lruEvict(self)/def mruEvict(self)
  - As explained previously you will have to implement these specialized removal functions yourself to remove content from the list based on when it was used (in this case added).
  - This does not need to return a value.
  - Notice that these operations are removals from the Linked List that we have done in the Module 5 Hands On session

- def find(self, cid)
  - This function should be able to find and return a contentItem object from a cache. If it is not found, it should return None.

```
>>> cache=Cache()
>>> content1 = ContentItem(1006, 170, "Content-Type: 2", "110010")
>>> cache.insert(content1, 'mru')
'INSERTED: CONTENT ID: 1006 SIZE: 170 HEADER: Content-Type: 2 CONTENT: 110010'
>>> cache.hierarchy[2].find(1006)
CONTENT ID: 1006 SIZE: 170 HEADER: Content-Type: 2 CONTENT: 110010
```

  - If the content is found in the list, the object is moved at the beginning of the list (modifies the list)

```
>>> cache = Cache()
>>> content1 = ContentItem(1006, 100, "Content-Type: 2", "110010")
>>> content2 = ContentItem(1004, 30, "Content-Type: 2", "11111")
>>> content3 = ContentItem(1003, 40, "Content-Type: 2", "11111")
>>> cache.insert(content1,'mru')
>>> cache.insert(content2,'mru')
>>> cache.insert(content3,'mru')
>>> cache
…
L3 CACHE:
REMAINING SPACE:0
ITEMS:2
LIST:
[CONTENT ID: 1003 SIZE: 40 HEADER: Content-Type: 2 CONTENT: 11111]
[CONTENT ID: 1004 SIZE: 30 HEADER: Content-Type: 2 CONTENT: 110010]
[CONTENT ID: 1006 SIZE: 100 HEADER: Content-Type: 2 CONTENT: 110010]
```

0	1	2
CacheList(200)	CacheList(200)	CacheList(200) content3 ↓ content2 ↓ content1

```
>>> cache.hierarchy[2].find(1004)
CONTENT ID: 1004 SIZE: 30 HEADER: Content-Type: 2 CONTENT: 110010]
>>> cache
…
L3 CACHE:
REMAINING SPACE:0
ITEMS:2
LIST:
[CONTENT ID: 1004 SIZE: 30 HEADER: Content-Type: 2 CONTENT: 110010]
[CONTENT ID: 1003 SIZE: 40 HEADER: Content-Type: 2 CONTENT: 11111]
[CONTENT ID: 1006 SIZE: 100 HEADER: Content-Type: 2 CONTENT: 110010]
```

0	1	2
CacheList(200)	CacheList(200)	CacheList(200) content2 ↓ content3 ↓ content1

- def update(self, cid, content)
  - This function should be able to find and return a content object from a cache. If it is not found, it should return None.
  - Similar to find, if the content is updated in the list, the object is moved at the beginning of the list (modifies the list)

```
>>> cache
...
L3 CACHE:
REMAINING SPACE:0
ITEMS:2
LIST:
[CONTENT ID: 1004 SIZE: 30 HEADER: Content-Type: 2 CONTENT: 11111]
[CONTENT ID: 1006 SIZE: 70 HEADER: Content-Type: 2 CONTENT: 110010]
>>> content3 = ContentItem(1006, 70, "Content-Type: 2", "random content")
>>> cache.hierarchy[2].update(1006,content3)
'UPDATED: CONTENT ID: 1006 SIZE: 70 HEADER: Content-Type: 2 CONTENT: random content'
>>> cache
...
L3 CACHE:
REMAINING SPACE:0
ITEMS:2
LIST:
[CONTENT ID: 1006 SIZE: 70 HEADER: Content-Type: 2 CONTENT: random content]
[CONTENT ID: 1004 SIZE: 30 HEADER: Content-Type: 2 CONTENT: 11111]
```

- def clear(self)
  - This function should remove every single item in a cache, and reset the cache's size and number of items. When all items are removed, it should return a message indicating so. This is shown in the doctests.

As a general note, please keep all __str__ and __repr__ functions intact. They have been written to visualize the content in a consistent and easy to read way. **Note that this is just the legible representation for the personalized objects, so you should not use their output in any of the operations you were asked to complete**

Deliverables

- Submit your HW4.py file to the HW4 Gradescope assignment before the due date.

Doctests (please note that the <blankline> does not need to show up in your code, it should just be a blank line). This doctest can be found in the HW4_doctest.py file on Canvas in the HW4 assignment

TIP: Execute the entire doctest only when ready to test out all your code. For debugging, it is best to perform separate calls in the Python Shells

```
    >>> cache = Cache()
    >>> content1 = ContentItem(1000, 10, "Content-Type: 0", "0xA")
    >>> content2 = ContentItem(1003, 13, "Content-Type: 0", "0xD")
    >>> content3 = ContentItem(1008, 242, "Content-Type: 0", "0xF2")

    >>> content4 = ContentItem(1004, 50, "Content-Type: 1", "110010")
    >>> content5 = ContentItem(1001, 51, "Content-Type: 1", "110011")
    >>> content6 = ContentItem(1007, 155, "Content-Type: 1", "10011011")

    >>> content7 = ContentItem(1005, 18, "Content-Type: 2", "<html><p>'CMPSC132'</p></html>")
    >>> content8 = ContentItem(1002, 14, "Content-Type: 2", "<html><h2>'PSU'</h2></html>")
    >>> content9 = ContentItem(1006, 170, "Content-Type: 2", "<html><button>'Click Me'</button></html>")

    >>> cache.insert(content1, 'lru')
    'INSERTED: CONTENT ID: 1000 SIZE: 10 HEADER: Content-Type: 0 CONTENT: 0xA'
    >>> cache.insert(content2, 'lru')
    'INSERTED: CONTENT ID: 1003 SIZE: 13 HEADER: Content-Type: 0 CONTENT: 0xD'
    >>> cache.insert(content3, 'lru')
    'Insertion not allowed. Content size is too large.'

    >>> cache.insert(content4, 'lru')
    'INSERTED: CONTENT ID: 1004 SIZE: 50 HEADER: Content-Type: 1 CONTENT: 110010'
    >>> cache.insert(content5, 'lru')
    'INSERTED: CONTENT ID: 1001 SIZE: 51 HEADER: Content-Type: 1 CONTENT: 110011'
    >>> cache.insert(content6, 'lru')
    'INSERTED: CONTENT ID: 1007 SIZE: 155 HEADER: Content-Type: 1 CONTENT: 10011011'

    >>> cache.insert(content7, 'lru')
    "INSERTED: CONTENT ID: 1005 SIZE: 18 HEADER: Content-Type: 2 CONTENT: <html><p>'CMPSC132'</p></html>"
    >>> cache.insert(content8, 'lru')
    "INSERTED: CONTENT ID: 1002 SIZE: 14 HEADER: Content-Type: 2 CONTENT: <html><h2>'PSU'</h2></html>"
    >>> cache.insert(content9, 'lru')
    "INSERTED: CONTENT ID: 1006 SIZE: 170 HEADER: Content-Type: 2 CONTENT: <html><button>'Click
Me'</button></html>"
    >>> cache
    L1 CACHE:
    REMAINING SPACE:177
    ITEMS:2
    LIST:
    [CONTENT ID: 1003 SIZE: 13 HEADER: Content-Type: 0 CONTENT: 0xD]
    [CONTENT ID: 1000 SIZE: 10 HEADER: Content-Type: 0 CONTENT: 0xA]
    <BLANKLINE>
    <BLANKLINE>
    L2 CACHE:
    REMAINING SPACE:45
    ITEMS:1
    LIST:
    [CONTENT ID: 1007 SIZE: 155 HEADER: Content-Type: 1 CONTENT: 10011011]
    <BLANKLINE>
    <BLANKLINE>
    L3 CACHE:
    REMAINING SPACE:16
    ITEMS:2
    LIST:
    [CONTENT ID: 1006 SIZE: 170 HEADER: Content-Type: 2 CONTENT: <html><button>'Click Me'</button></html>]
    [CONTENT ID: 1002 SIZE: 14 HEADER: Content-Type: 2 CONTENT: <html><h2>'PSU'</h2></html>]
    <BLANKLINE>
    <BLANKLINE>
    <BLANKLINE>
    >>> cache.hierarchy[0].clear()
    'Cleared cache!'
    >>> cache.hierarchy[1].clear()
    'Cleared cache!'
    >>> cache.hierarchy[2].clear()
    'Cleared cache!'
    >>> cache
    L1 CACHE:
    REMAINING SPACE:200
```

```
    ITEMS:0
    LIST:
    <BLANKLINE>
    <BLANKLINE>
    L2 CACHE:
    REMAINING SPACE:200
    ITEMS:0
    LIST:
    <BLANKLINE>
    <BLANKLINE>
    L3 CACHE:
    REMAINING SPACE:200
    ITEMS:0
    LIST:
    <BLANKLINE>
    <BLANKLINE>
    <BLANKLINE>
    >>> cache.insert(content1, 'mru')
    'INSERTED: CONTENT ID: 1000 SIZE: 10 HEADER: Content-Type: 0 CONTENT: 0xA'
    >>> cache.insert(content2, 'mru')
    'INSERTED: CONTENT ID: 1003 SIZE: 13 HEADER: Content-Type: 0 CONTENT: 0xD'
    >>> cache.retrieveContent(content1)
    CONTENT ID: 1000 SIZE: 10 HEADER: Content-Type: 0 CONTENT: 0xA
    >>> cache.retrieveContent(content2)
    CONTENT ID: 1003 SIZE: 13 HEADER: Content-Type: 0 CONTENT: 0xD
    >>> cache.retrieveContent(content3)
    'Cache miss!'

    >>> cache.insert(content5, 'lru')
    'INSERTED: CONTENT ID: 1001 SIZE: 51 HEADER: Content-Type: 1 CONTENT: 110011'
    >>> cache.insert(content6, 'lru')
    'INSERTED: CONTENT ID: 1007 SIZE: 155 HEADER: Content-Type: 1 CONTENT: 10011011'
    >>> cache.insert(content4, 'lru')
    'INSERTED: CONTENT ID: 1004 SIZE: 50 HEADER: Content-Type: 1 CONTENT: 110010'


    >>> cache.insert(content7, 'mru')
    "INSERTED: CONTENT ID: 1005 SIZE: 18 HEADER: Content-Type: 2 CONTENT: <html><p>'CMPSC132'</p></html>"
    >>> cache.insert(content8, 'mru')
    "INSERTED: CONTENT ID: 1002 SIZE: 14 HEADER: Content-Type: 2 CONTENT: <html><h2>'PSU'</h2></html>"
    >>> cache.insert(content9, 'mru')
    "INSERTED: CONTENT ID: 1006 SIZE: 170 HEADER: Content-Type: 2 CONTENT: <html><button>'Click
Me'</button></html>"
    >>> cache
    L1 CACHE:
    REMAINING SPACE:177
    ITEMS:2
    LIST:
    [CONTENT ID: 1003 SIZE: 13 HEADER: Content-Type: 0 CONTENT: 0xD]
    [CONTENT ID: 1000 SIZE: 10 HEADER: Content-Type: 0 CONTENT: 0xA]
    <BLANKLINE>
    <BLANKLINE>
    L2 CACHE:
    REMAINING SPACE:150
    ITEMS:1
    LIST:
    [CONTENT ID: 1004 SIZE: 50 HEADER: Content-Type: 1 CONTENT: 110010]
    <BLANKLINE>
    <BLANKLINE>
    L3 CACHE:
    REMAINING SPACE:12
    ITEMS:2
    LIST:
    [CONTENT ID: 1006 SIZE: 170 HEADER: Content-Type: 2 CONTENT: <html><button>'Click Me'</button></html>]
    [CONTENT ID: 1005 SIZE: 18 HEADER: Content-Type: 2 CONTENT: <html><p>'CMPSC132'</p></html>]
    <BLANKLINE>
    <BLANKLINE>
    <BLANKLINE>

    >>> cache.clear()
    'Cache cleared!'
    >>> contentA = ContentItem(2000, 52, "Content-Type: 2", "GET https://www.pro-football-
reference.com/boxscores/201802040nwe.htm HTTP/1.1")
    >>> contentB = ContentItem(2001, 76, "Content-Type: 2", "GET https://giphy.com/gifs/93lCI4D0murAszeyA6/html5
HTTP/1.1")
    >>> contentC = ContentItem(2002, 11, "Content-Type: 2", "GET
https://media.giphy.com/media/YN7akkfUNQvT1zEBhO/giphy-downsized.gif HTTP/1.1")
    >>> cache.insert(contentA, 'lru')
```

```
    'INSERTED: CONTENT ID: 2000 SIZE: 52 HEADER: Content-Type: 2 CONTENT: GET https://www.pro-football-
reference.com/boxscores/201802040nwe.htm HTTP/1.1'
    >>> cache.insert(contentB, 'lru')
    'INSERTED: CONTENT ID: 2001 SIZE: 76 HEADER: Content-Type: 2 CONTENT: GET
https://giphy.com/gifs/93lCI4D0murAszeyA6/html5 HTTP/1.1'
    >>> cache.insert(contentC, 'lru')
    'INSERTED: CONTENT ID: 2002 SIZE: 11 HEADER: Content-Type: 2 CONTENT: GET
https://media.giphy.com/media/YN7akkfUNQvT1zEBhO/giphy-downsized.gif HTTP/1.1'
    >>> cache.hierarchy[2]
    REMAINING SPACE:61
    ITEMS:3
    LIST:
    [CONTENT ID: 2002 SIZE: 11 HEADER: Content-Type: 2 CONTENT: GET
https://media.giphy.com/media/YN7akkfUNQvT1zEBhO/giphy-downsized.gif HTTP/1.1]
    [CONTENT ID: 2001 SIZE: 76 HEADER: Content-Type: 2 CONTENT: GET
https://giphy.com/gifs/93lCI4D0murAszeyA6/html5 HTTP/1.1]
    [CONTENT ID: 2000 SIZE: 52 HEADER: Content-Type: 2 CONTENT: GET https://www.pro-football-
reference.com/boxscores/201802040nwe.htm HTTP/1.1]
    <BLANKLINE>
    <BLANKLINE>
    >>> cache.retrieveContent(contentC)
    CONTENT ID: 2002 SIZE: 11 HEADER: Content-Type: 2 CONTENT: GET
https://media.giphy.com/media/YN7akkfUNQvT1zEBhO/giphy-downsized.gif HTTP/1.1
    >>> cache.hierarchy[2]
    REMAINING SPACE:61
    ITEMS:3
    LIST:
    [CONTENT ID: 2002 SIZE: 11 HEADER: Content-Type: 2 CONTENT: GET
https://media.giphy.com/media/YN7akkfUNQvT1zEBhO/giphy-downsized.gif HTTP/1.1]
    [CONTENT ID: 2001 SIZE: 76 HEADER: Content-Type: 2 CONTENT: GET
https://giphy.com/gifs/93lCI4D0murAszeyA6/html5 HTTP/1.1]
    [CONTENT ID: 2000 SIZE: 52 HEADER: Content-Type: 2 CONTENT: GET https://www.pro-football-
reference.com/boxscores/201802040nwe.htm HTTP/1.1]
    <BLANKLINE>
    <BLANKLINE>
    >>> cache.retrieveContent(contentA)
    CONTENT ID: 2000 SIZE: 52 HEADER: Content-Type: 2 CONTENT: GET https://www.pro-football-
reference.com/boxscores/201802040nwe.htm HTTP/1.1
    >>> cache.hierarchy[2]
    REMAINING SPACE:61
    ITEMS:3
    LIST:
    [CONTENT ID: 2000 SIZE: 52 HEADER: Content-Type: 2 CONTENT: GET https://www.pro-football-
reference.com/boxscores/201802040nwe.htm HTTP/1.1]
    [CONTENT ID: 2002 SIZE: 11 HEADER: Content-Type: 2 CONTENT: GET
https://media.giphy.com/media/YN7akkfUNQvT1zEBhO/giphy-downsized.gif HTTP/1.1]
    [CONTENT ID: 2001 SIZE: 76 HEADER: Content-Type: 2 CONTENT: GET
https://giphy.com/gifs/93lCI4D0murAszeyA6/html5 HTTP/1.1]
    <BLANKLINE>
    <BLANKLINE>
    >>> cache.retrieveContent(contentC)
    CONTENT ID: 2002 SIZE: 11 HEADER: Content-Type: 2 CONTENT: GET
https://media.giphy.com/media/YN7akkfUNQvT1zEBhO/giphy-downsized.gif HTTP/1.1
    >>> cache.hierarchy[2]
    REMAINING SPACE:61
    ITEMS:3
    LIST:
    [CONTENT ID: 2002 SIZE: 11 HEADER: Content-Type: 2 CONTENT: GET
https://media.giphy.com/media/YN7akkfUNQvT1zEBhO/giphy-downsized.gif HTTP/1.1]
    [CONTENT ID: 2000 SIZE: 52 HEADER: Content-Type: 2 CONTENT: GET https://www.pro-football-
reference.com/boxscores/201802040nwe.htm HTTP/1.1]
    [CONTENT ID: 2001 SIZE: 76 HEADER: Content-Type: 2 CONTENT: GET
https://giphy.com/gifs/93lCI4D0murAszeyA6/html5 HTTP/1.1]
    <BLANKLINE>
    <BLANKLINE>
    >>> contentD = ContentItem(2002, 11, "Content-Type: 2", "GET
https://media.giphy.com/media/YN7akkfUNQvT1zEBhO/giphy-downsized.gif HTTP/1.1")
    >>> cache.insert(contentD, 'lru')
    'Insertion of content item 2002 not allowed. Content already in cache.'
    >>> contentE = ContentItem(2000, 52, "Content-Type: 2", "GET https://www.pro-football-
reference.com/boxscores/201801210phi.htm HTTP/1.1")
    >>> cache.updateContent(contentE)
    'UPDATED: CONTENT ID: 2000 SIZE: 52 HEADER: Content-Type: 2 CONTENT: GET https://www.pro-football-
reference.com/boxscores/201801210phi.htm HTTP/1.1'
    >>> cache.hierarchy[2]
    REMAINING SPACE:61
    ITEMS:3
    LIST:
```

[CONTENT ID: 2000 SIZE: 52 HEADER: Content-Type: 2 CONTENT: GET https://www.pro-football-reference.com/boxscores/201801210phi.htm HTTP/1.1]
[CONTENT ID: 2002 SIZE: 11 HEADER: Content-Type: 2 CONTENT: GET https://media.giphy.com/media/YN7akkfUNQvT1zEBhO/giphy-downsized.gif HTTP/1.1]
[CONTENT ID: 2001 SIZE: 76 HEADER: Content-Type: 2 CONTENT: GET https://giphy.com/gifs/93lCI4D0murAszeyA6/html5 HTTP/1.1]
<BLANKLINE>
<BLANKLINE>