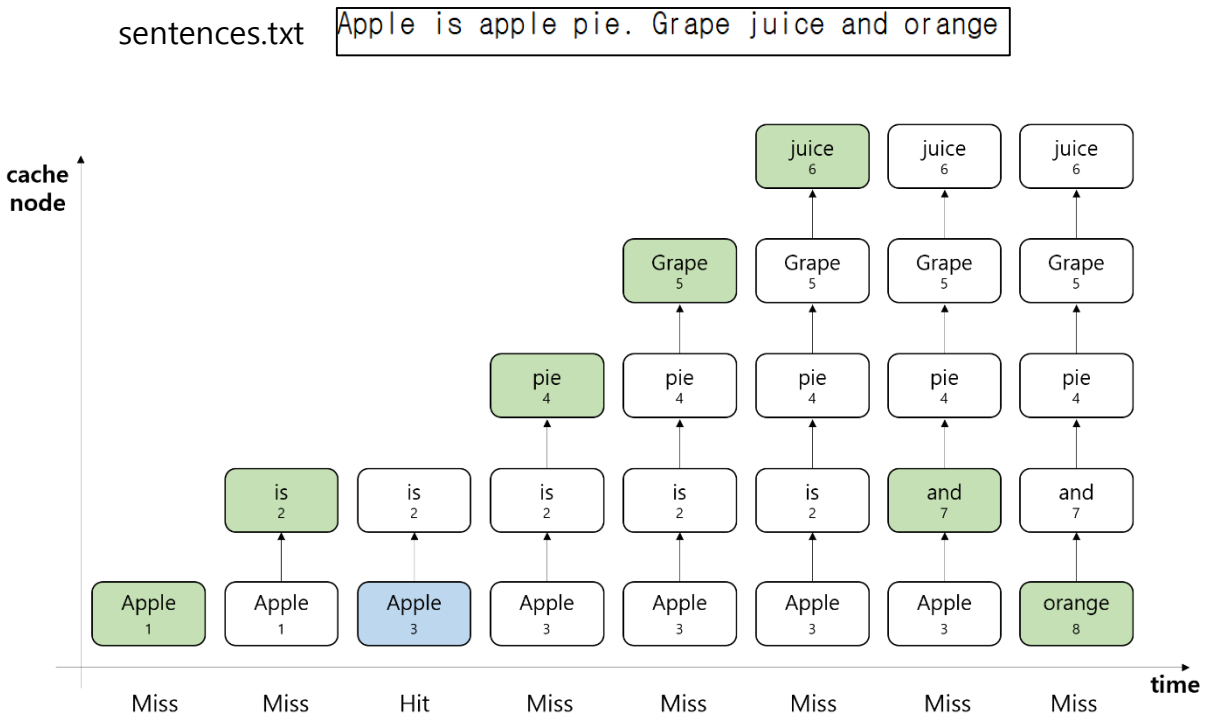
Object Oriented Programming

assignment 3-2

ASSIGNMENT 3-2

Overview

- Implement a simple cache system which stores words read from the file 'sentences.txt' using LRU(Least recent used) algorithm. Cache has a structure of single linked list composed of maximum five 'CacheNode' classes.



Cache structure

- Cache has a structure of linked list composed of maximum five 'CacheNode' classes
- 'CacheNode' and 'CacheManager' Classes must be used for a given purpose

- Class CacheNode
  - The node of cache linked list
  - Member variables(Another data type can be used)
    - Char*/String Word
      - Stores the word
    - int Timestamp
      - Stores the timestamp value of the word
    - CacheNode* NextNode
      - Stores address of next node of the cache linked list

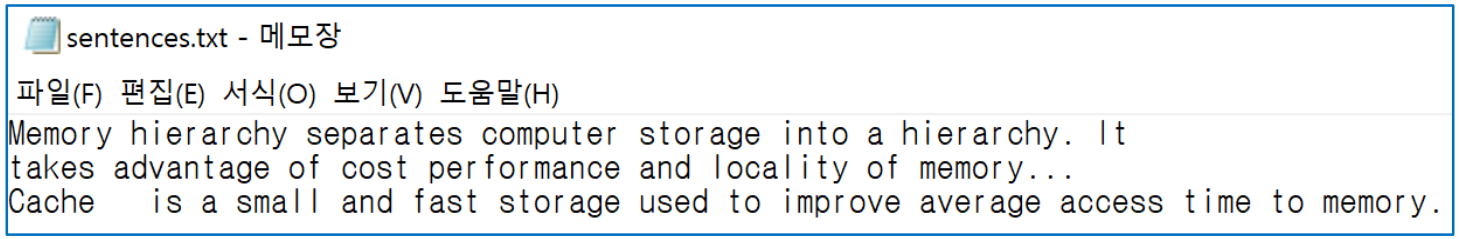
Cache structure

- Class CacheManager
  - Manages cache system
  - Member variables
    - CacheNode* HeadNode
      - Address of the head node of the linked list
  - Member functions
    - Check whether the word is in the cache
    - Then performs an operation according to the determination result(hit or miss)

Caching rules

- Accesses the words stored in the input file in order and caches each word according to the following method
- At the beginning of program execution, the cache (linked list) is empty.
- When caching, words are not case sensitive.

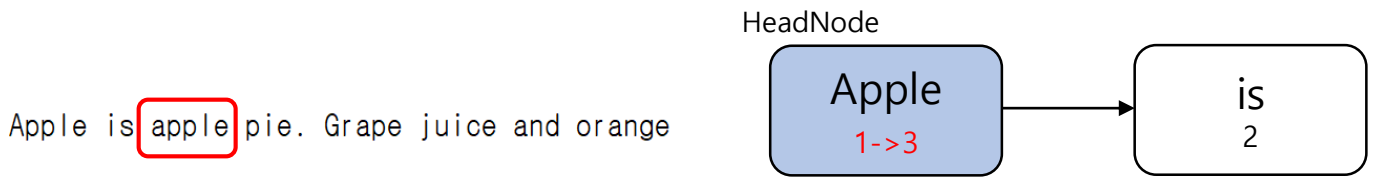
- A word in an input file consists of one or more consecutive alphabets (upper and lower case).
- Words in an input file can be separated by one or more consecutive non-alphabet characters (space character, new line character, period).



33 words in file : Memory, hierarchy, separates, computer, storage, into, a, hierarchy, It, takes, advantage, of, cost, performance, and, locality, of, memory, Cache, is, a, small, and, fast, storage, used, to, improve, average, access, time, to, memory

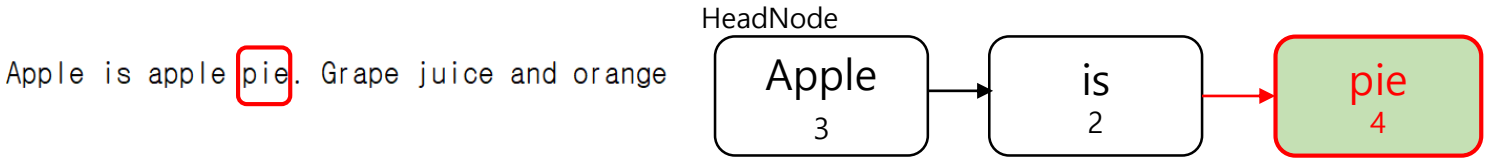
Caching rules

- If the same word as the word from input file exists in linked list,
  - Print message as 'Hit'
  - Update variable 'timestamp' of that node to the current timestamp value
    - Timestamp is the sequence number of words when it occurs in the file(Start at 1 and increment by 1)

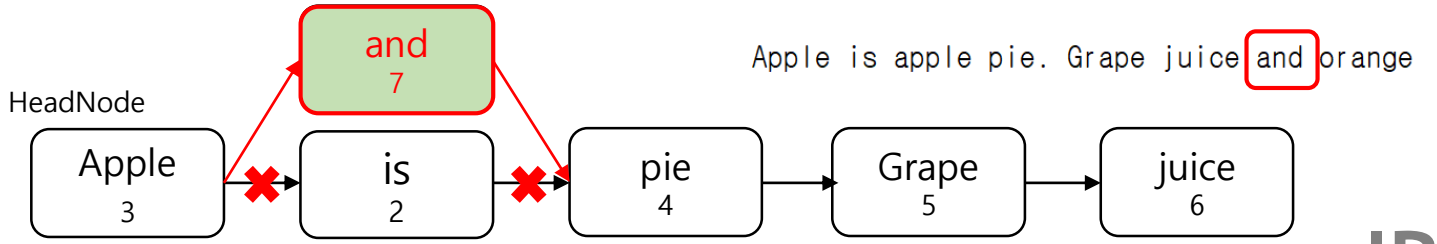


Caching rules

- Else,
  - Print message as 'Miss'
  - Create an object of 'CacheNode'
  - Save the word read from input file in a variable 'Word' of the new node and save the current timestamp value in a variable 'Timestamp' of the new node
  - If cache is not full,
    - and add it to the end of the linked list



  - If cache is full,
    - Replace the node that stores the word that occurred the longest ago in the linked list with a new node. (LRU algorithm)



Output example

- Print Hit or Miss each time
- After every caching, print out all the words with timestamp in the cache

```
Miss Apple:1
Miss Apple:1  is:2
Hit  Apple:3  is:2
Miss Apple:3  is:2  pie:4
Miss Apple:3  is:2  pie:4  Grape:5
Miss Apple:3  is:2  pie:4  Grape:5  juice:6
Miss Apple:3  and:7  pie:4  Grape:5  juice:6
Miss orange:8  and:7  pie:4  Grape:5  juice:6
```