Tennessee TECH

CSC 2111: Lab 10

HYBRID ADT

In this lab, you will implement an ADT that requires elements of two very different ADTs discussed in class.

Hybrid ADT

- bool isEmpty()
- void enqueue(T* item) //add and remove items in **FIFO** order
- T* dequeue()
- ListDoublyLinkedIterator<T>* iterator() //iterate over the items in **sorted** order

Complete the T* remove(DoubleNode<T>* curr) method in SLDL.

Easy Implementation The above ADT requires adds and removes in FIFO order. If this were all that was required, you could easily implement the ADT with a queue. However, the ADT also requires iteration over the elements in sorted order. This cannot be done with queue operations. Implement the Hybrid ADT by using **both** a QueueLinked and a SortedListDoublyLinked.

There is an efficiency issue with this implementation of the Hybrid ADT. When enqueue is called on the Hybrid ADT, the item is enqueued on the queue, an O(1) operation. The item must also be added to the sorted list, an O(n) operation, so the actual efficiency of the Hybrid ADT enqueue is O(n). Likewise, when an item is dequeued and removed, this is again O(1)/O(n), which yields O(n). There is nothing that can be done to improve the enqueue/add efficiency. However, it is possible to improve the efficiency class for the dequeue/removal operation to O(1).

Efficient, Complex Implementation Modify your Hybrid ADT implementation so that the queue can hold DoubleNodes (**T = DoubleNode<T>**). When an item is added to the sorted list, the DoubleNode containing the data is returned. Place this DoubleNode on the queue. This way, when dequeueing from the queue, you have a direct reference to the DoubleNode in the SortedListDoublyLinked, and you can remove the item from the sorted list immediately. This requires SortedListDoublyLinked to have two special methods that you will need to call. In order to call them, you will need to move them from private to public in the header file.

- DoubleNode<T>* addDN(T* item) //add the item to the sorted list, and return a pointer to the DoubleNode containing the data just added
- T* remove(DoubleNode<T>* curr) //remove the DoubleNode specified from the sorted list, and return the pointer to its data

- The below class interfaces are defined in the namespace **CSC2110**.

  - String methods
    - String(const char* text) //constructor
    - void displayString()
    - int length()
    - const char* getText()
    - int a_to_i()
    - float a_to_f()
    - String* i_to_a(int number) //static method
    - String* f_to_a(float number) //static method
    - int find(char delimiter, int start) //find the index of a particular character
    - String* substr(int start, int end)
    - int compare(String* other)
    - char charAt(int index) //0-based
  - Tokens methods
    - Tokens(String* str, char delimiter) //destructor does not delete the individual tokens
    - String* getToken(int index)
    - int getNumTokens()
    - void displayTokens()
  - ReadFile methods
    - ReadFile(const char* file_name)
    - String* readLine()
    - bool eof()
    - void close()
  - WriteFile methods
    - WriteLine(String* file_name)
    - void writeLine(String* line)
    - void close()
  - Random methods (use getRandom first)
    - static Random* getRandom()
    - int getRandomInt(int lower, int upper)
    - float getRandomFloat(float lower, float upper)
  - Keyboard methods (use getKeyboard first)
    - static Keyboard* getKeyboard() //call this first
    - int readInt(string prompt)
    - int getValidatedInt(string prompt, int min, int max)
    - double readDouble(string prompt)
    - double getValidatedDouble(string prompt, double min, double max)
    - String* readString(string prompt)
  - Integer/Double methods

    - Integer(int val)/Double(double val)
    - int getValue()/double getValue()
- CD methods
    - CD(String* artist, String* title, int year, int rating, int num_tracks)
    - void displayCD() //display the current state of the CD
    - String* getKey()
    - void addSong(String* title, String* length)
    - static int compare_items(CD* one, CD* two) //define how to compare CDs (in this case, by title)
    - static int compare_keys(String* sk, CD* cd)
    - static ListArray* readCDs(const char* file_name) //read in all of the CDs from a text file
- Song methods
    - Song(String* title, String* length)
    - void displaySong()