Dept. of Computer Science & Software Eng., Concordia University
COMP 6481 --- Winter 2022

Programming and Problem Solving

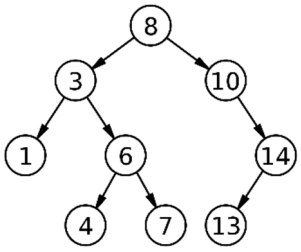
Assignment 3 --- Due Wednesday, April 13, 2022

Part I

Please read carefully: You must submit the answers to all the questions below. However, this part will not be marked. Nonetheless, failing to submit this part fully will result in you missing 50% of the total mark of the assignment.

Question 1

Write an efficient pseudo-code for a method named BottomUpLevelTraversal which is a variation of BST level order traversal. It is similar to the level order traversal; except, that nodes on deeper levels are visited before nodes on shallower levels. Your method must not modify the tree. Also, if the tree has n nodes, what is the running time of your new traversal method?



A sample tree where BottomUpLevelTraversal will print 4,7,13,1,6,14,3,10,8.

Question 2

You are given a hash table with size 7 and a hashing function h(k) = k%7. Output the state of the table after inserting the below mentioned values in that specific order when collisions are handled using:

a. Separate chaining

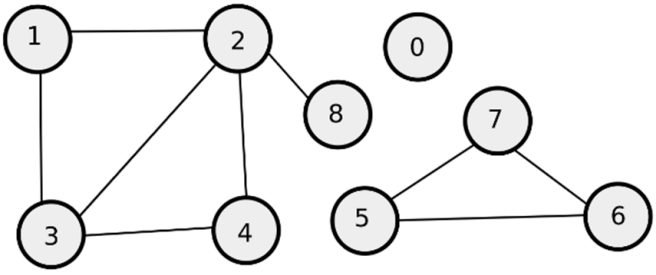
b. Linear probing

c. Double hashing using a second hash function h'(k) = 5 – (k % 5)

The values to be inserted are 19, 26, 13, 48, 17.

Question 3

Develop a **well-documented pseudo code** that determines number of connected components in any given graph. A graph is connected when there is a path between every pair of vertices. A graph G is said to be disconnected if there exist at least two nodes in G such that no path in G has those nodes as endpoints. A connected component (or just component) of a graph is a sub graph in which any two vertices are connected to each other by paths, and which is connected to no additional vertices in the super graph. For instance, the below mentioned graph will yield 3 as it has three connected components.

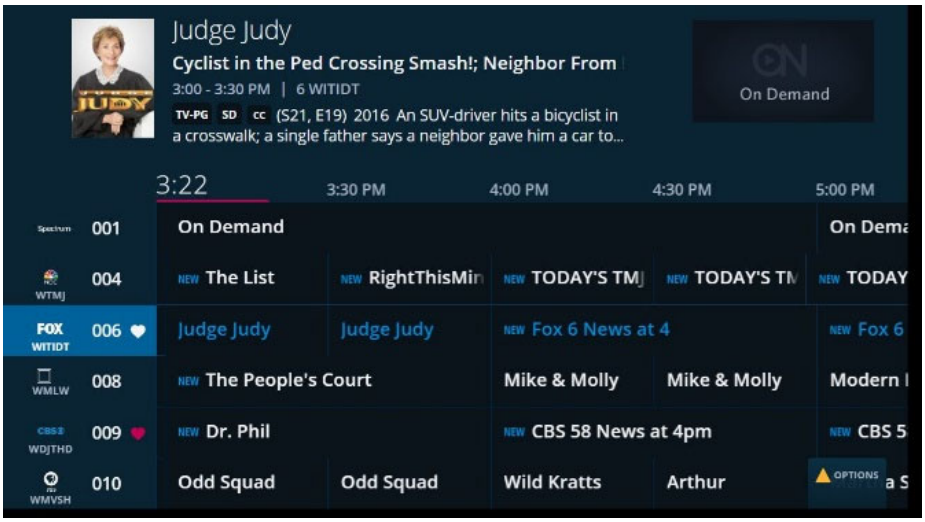


Part II

Purpose: The purpose of this assignment is to allow you practice Linked Lists.

"Television is an invention that permits you to be entertained in your living room by people you wouldn't have in your home."

-- David Frost



(TVGuide Image from spectrum.net)

Watching televison these days is pretty easy with the advent of internet tv. We no longer need a TV guide to keep track of which shows are when. We can record them to watch when we please or even better just stream them whenever and wherever. A few years ago, this was not the case. TVGuides were very much a necessity to know what's playing when. This would then allow people to choose the shows that they want to watch over the others.
As an example, CBS brodcasts Criminal Minds at 10pm and ABC brodcasts Shark Tank at the same time. As such, it is impossible to watch both the shows. On the other hand, ABC brodcasts Modern Family at 9.30pm and therefore, one can watch both Shark Tank as well as Modern Family.

In this assignment, you will design and implement a TVGuide which will determine if a user can watch a specific show based on shows he/she is currently watching. You are given two files, namely, TVGuide.txt containing information about various TV shows, and Interests.txt which contains information about the shows a user is interested in. You will parse these files to extract TV shows information and will produce an outcome for each of the show a user wants to watch. The outcome for each show could be one of the below mentioned options where X represents showID, S represents start time and E represents end time.

a) User can watch show X as he/she is not watching anything else during that time.
b) User can't watch show X as he/she is not finished with a show he/she is watching.
c) User can't watch show X as he/she will begin another show at the same time.

You can assume that every show will be minimum half an hour long and maximum one hour long. You can also assume that the shows will be listed in a sorted manner based on their start time. Interests.txt will contain a word "Watching" on first line, followed by showID of the shows he/she is currently watching. Another word "Wishlist" after the above information followed by the respective showIDs. TVGuide.txt will contain showID along with show name (one word seperated by _). Next two lines will have S and E indicating start time and end time respectively for this show. This set of information is repeated for all the available shows. A sample TVGuide.txt file is depicted in Figure 3a and a sample Interest.txt is depicted in Figure 3b. A detailed description of all the details that you have to implement for this assignment is made avaialble after these two figures.

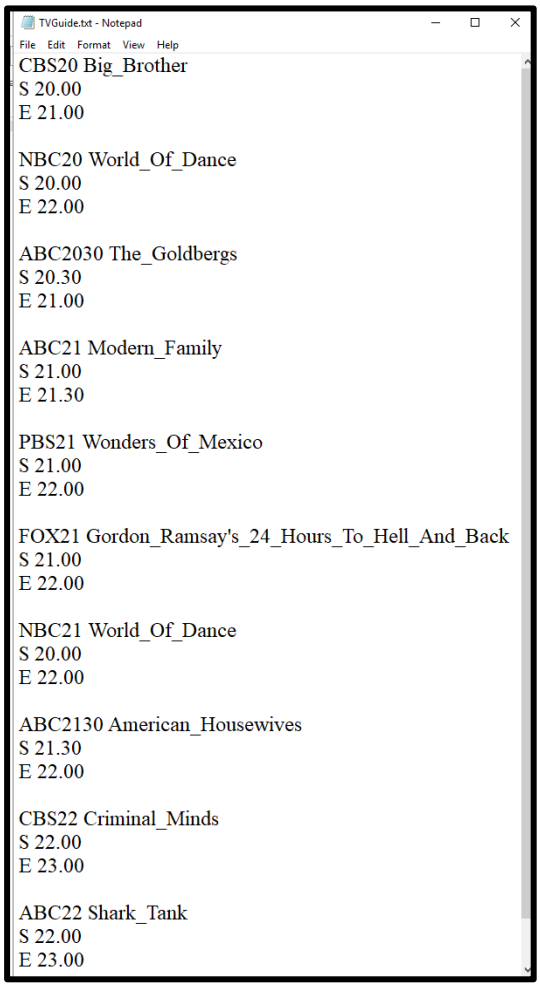


Figure 3a: Illustration of TVGuide.txt where S represents start time and E represents end time

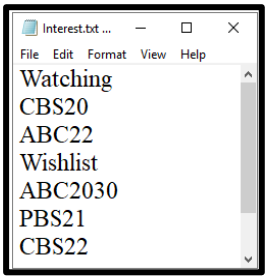


Figure 3b: Illustration of Request.txt

I) Create an interface named Watchable which has a method string isOnSameTime (Show S) where S is a object of type TVShow described in the next part.

II) The TVShow class has the following attributes: a showID (String type), a showName (String type), a startTime (double type), an endTime (double type). It is assumed that show name is always recorded as a single word (_ is used to combine multiple words). It is also assumed that no two TV shows can have the exact same showID.
You are required to write the implementation of the Show class. Besides the usual mutator and accessor methods (i.e. getShowID(), setShowName()) the class must also have the following:

a) Parameterized constructor that accepts four values and initializes showID, showName, startTime, and endTime to these passed values;
b) Copy constructor, which takes in two parameters, a show object and a String value. The newly created object will be assigned all the attributes of the passed object, with the exception of the

showID. showID is assigned the value passed as the second parameter to the constructor. It is always assumed that this value will correspond to the unique showID rule;

c) clone() method. This method will prompt the user to enter a new showID, then creates and returns a clone of the calling object with the exception of the showID, which is assigned the value entered by the user;

d) Additionally, the class should have a toString() and an equals() methods. Two shows are equal if they have the same attributes, with the exception of the showID, which could be different.

e) This class needs to implement the interface from part I. The method isOnSameTime that takes in another TVShow object S and should return "Same time", "Different time", or "Some Overlap" depending on the times of two TVShow objects.

III) The **ShowList** class has the following:

(a) An inner class called ShowNode. This class has the following:
   i. Two private attributes: an object of TVShow and a pointer to a ShowNode object.
   ii. A default constructor, which assigns both attributes to null.
   iii. A parameterized constructor that accepts two parameters, a TVShow object and a ShowNode object, then initializes the attributes accordingly.
   iv. A copy constructor.
   v. A clone() method
   vi. Other mutator and accessor methods.

(b) A private attribute called head, which should point to the first node in this list object.

(c) A private attribute called size, which always indicates current size of the list (number of nodes in the list);

(d) A default constructor, which creates an empty list.

(e) A copy constructor, which accepts a ShowList object and creates a copy of it.

(f) A method called addToStart(), which accepts one parameter, a TVShow object and then creates a node with that passed object and inserts this node at the head of the list;

(g) A method called insertAtIndex(), which accepts two parameters, an object from TVShow class, and an integer representing an index. If the index is not valid (a valid index must have a value between 0 and size-1), the method must throw a NoSuchElementException and terminate the program. If the index is valid, then the method creates a node with the passed TVShow object and inserts this node at the given index. The method must properly handle all special cases.

(h) A method called deleteFromIndex(), which accepts one integer parameter representing an index. Again, if the index is not valid, the method must throw a NoSuchElementException and terminate the program. Otherwise, the node pointed by that index is deleted from the list. The method must properly handle all special cases.

(i) A method called deleteFromStart(), which deletes first node in the list (the one pointed by head). All special cases must be properly handled.

(j) A method called replaceAtIndex(), which accepts two parameters, a TVShow object, and an integer representing an index. If the index is not valid, the method simply returns; otherwise, the object in the node at the passed index is to be replaced by the passed object.

(k) A method called find(), which accepts one parameter of type String representing a showID. The method then searches the list for a ShowNode with that showID. If such an object is found, then the method returns a pointer to that ShowNode; otherwise, method returns null. The method must keep track of how many iterations were made before the search finally finds the course or concludes that it is not in the list.

(l) A method called contains(), which accepts one parameter of type String representing a showID. It returns true if a course with that showID is in the list; otherwise, the method returns false.

(m) A method called equals(), which accepts one parameter of type ShowList. The method returns true if the two lists contain similar shows; otherwise, the method returns false. Recall that two TVShow objects are equal if they have the same values except for the showID, which can, and is expected to be, different.

A sample ShowList is demonstrated in Figure 4.

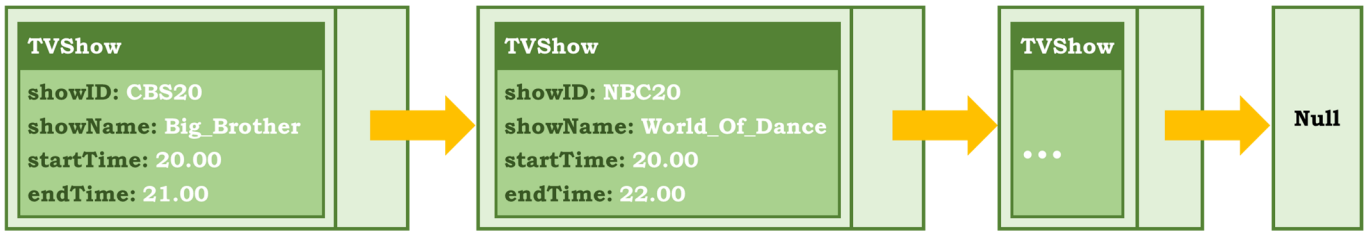


Figure 4: A sample ShowList (LinkedList implementation)

Finally, here are some general rules that you must consider when implementing the above methods:

-Whenever a node is added or deleted, the list size must be adjusted accordingly.
-All special cases must be handled, whether the method description explicitly states that or not.
-All clone () and copy constructors must perform a deep copy; no shallow copies are allowed.
-If any of your methods allows a privacy leak, you must clearly place a comment at the beginning of the method 1) indicating that this method may result in a privacy leak 2) explaining reason behind the privacy leak. Please keep in mind that you are not required to implement these proposals.

IV) You must write a public class called **ProcessWishlist**. In the main() method, you must do the following:

(a) Create at least two empty lists from the ShowList class (needed for copy constructor III (e)).
(b) Open the TVGuide.txt file and read its contents line by line. Use these records to initialize one of the ShowList objects you created above. You can use the addToStart () method to insert the read objects into the list. However, the list should not have any duplicate records, so if the input file has duplicate entries, which is the case in the file provided with the assignment for instance, your code must handle this case so that each record is inserted in the list only once.
(c) Open Interest.txt and create ArrayLists from the contents then iterate through each of the shows the user wants to watch. Process each of the shows and print the outcome whether user will be able to watch it or not. A sample output for a file in Figure 3b is mentioned below. Your program should ask for the file names as your program will be tested against similar input files.
(d) Prompt the user to enter a few showIDs and search the list that you created from the input file for these values. Make sure to display the number of iterations performed.
(e) Following that, you must create enough objects to test each of the constructors/methods of your classes. The details of this part are left as open to you. You can do whatever you wish if your methods are being tested including some of the special cases.

```
User can't watch show ABC2030 as he/she is not finished with a show he/she is watching.
User can watch show PBS21 as he/she is not watching anything else during that time.
User can't watch show CBS22 as he/she will begin another show at the same time.
```

Figure 5: A sample outcome of the wishlist from Figure 3b

Some general information:

a. You should open and close the TVGuide.txt file only once; a better mark will be given for that;
b. Do not use any external libraries or existing software to produce what is needed; that will directly result in a 0 mark!

Again, your program must work for any input files. The files provided with this assignment are only a possible version, and must not be considered as the general case when writing your code.

SUBMISSION INSTRUCTIONS

Submission format: All assignment-related submissions must be adequately archived in a ZIP file using your ID(s) and last name(s) as file name. The submission itself must also contain your name(s) and student ID(s). Use your "official" name only – no abbreviations/nick names; capitalize the usual "last" name. Inappropriate submissions will be heavily penalized. **IMPORTANT:** For Part II of the assignment, a demo for about 5 to 10 minutes will take place with the marker. You **must** attend the demo and be able to explain their program to the marker. The schedule of the demos will be determined and announced by the markers, and students must reserve a time slot for the demo (only one time slot per group).

Now, please read very carefully:

- **If you fail to demo, a zero mark is assigned regardless of your submission.**
- **If you book a demo time, and do not show up, for whatever reason, you will be allowed to reschedule a second demo but a penalty of 50% will be applied.**
- **Failing to demo at the second appointment will result in zero marks and no more chances will be given under any conditions.**

EVALUATION CRITERIA

IMPORTANT: **Part I** must fully be submitted. Failure to submit that part will cost 50% of the total marks of the assignment!

Total	**10 pts**
Documentations	**1 pt**
JavaDoc documentations	1 pt
Tasks	**9 pts**
Design and Correctness of Classes	4 pts
Proper and Sufficient Testing of Your Methods	2 pts
Privacy Leak Comments and Proposals to Avoid Them	1 pts
Grader QnA	2 pts