Specification for Assignment 2 of 4

IMPORTANT SUBMISSION INSTRUCTIONS

You will be uploading your submission using the assignment server. The link is posted to cuLearn. A short video is posted there also in case you require instructions. You may submit as many times as you wish, but must wait at least 5 minutes between submissions (to protect the server). Your highest mark is kept. It would help us out if you try the upload early, even using just the unmodified assignment skeleton, that way potential problems can be found before the deadline. It will also ensure that YOU know how to submit your assignment well before the deadline. If your attempt to obtain your secret code is unsuccessful, please email me at darrylhill@cunet.carleton.ca. This may happen if you have registered late, etc.

You must adhere to the following rules *(as your submission will be subjected to automatic marking system):*

- ***Download*** *the compressed file* ***"comp2402a2.zip"*** *from cuLearn.*
- ***Retain the directory structure*** *(i.e., if you find a file in subfolder "comp2402a2", you must not remove it).*
- ***Retain the package directives*** *(i.e., if you see "package comp2402a2;" in a file, you must not remove it).*
- ***Do not rename any of the methods already present*** *in the files provided.*
- ***Do not change the visibility of any of the methods provided*** *(e.g., do not change private to public).*
- ***Do not change the main method of any class****; on occasion the main method provided will use command line arguments and/or open input and output files – you must not change this behaviour).*
- ***Upload*** *a compressed file* ***"comp2402a2.zip"*** *to the assignment server to submit your assignment as receive your mark immediately* ***(highest mark of all submissions will be your assignment mark).***

Please also note that ***your code may be marked for efficiency as well as correctness*** *– this is accomplished by placing a hard limit on the amount of time your code wil be permitted for execution. If you select/apply your data structures correctly, your code will easily execute within the time limit, but* ***if your choice or usage of a data structure is incorrect****, your code may be* ***judged to be too slow*** *and* ***it may receive a grade of zero****.*

You are expected to ***demonstrate good programming practices at all times*** *(e.g., choosing appropriate variable names, provide comments in your code, etc.) and* ***your code may be penalized if it is poorly written****. The server won't judge this, but in case of discrepancies, this will be evaluated.*

Instructions

Start by downloading the comp2402a2 Zip file from cuLearn, which contains a skeleton of the code you need to write. This assignment is about building data structures. The folder contains three files (java classes); BlockedList.java, SSList.java, and Factory.java. You are to complete BlockedList.java and SSList.java according to the specification provided below. Factory.java is a workaround class to build generic arrays and should not be modified.

Specification for Assignment 2 of 4

Part 1 of 2 – Blocked List

For this question you must implement a **BlockedList** class that implements the **List** interface. You may use any of the classes in JCF or in the textbook code. The constructor for this class takes an integer block size **b** and the implementation should have the following performance characteristics:

a) **get(i)** and **set(i,x)** should run in **O(1)** time per operation

b) **add(i,x)** and **remove(i)** should run in **O(b+ min{i, n-i}/b)** amortized time per operation.

Any solution matching this specification is acceptable. However, the runtime would suggest a data structure that contains other data structures, or "blocks" of size at most **b**. For one possible solution, recall that an array backed Deque (an **ArrayDeque** in the textbook, or a **Deque** in the JCF) implements **add(i,x)** and **remove(i)** in **O(min{i, n-i})** amortized time per operation. We want a data structure with performance characteristics similar to an **ArrayDeque** (within a factor of **b)**. We might therefore consider an **ArrayDeque** of data structures. It is then left to you to determine the inner data structure. Also note, if we choose a block size of **b=1**, then we simply have an **ArrayDeque** running in **O(b+ min{i, n-i}/b) = O(min{i, n-i})** amortized time per operation, as required, regardless of the inner data structure chosen.

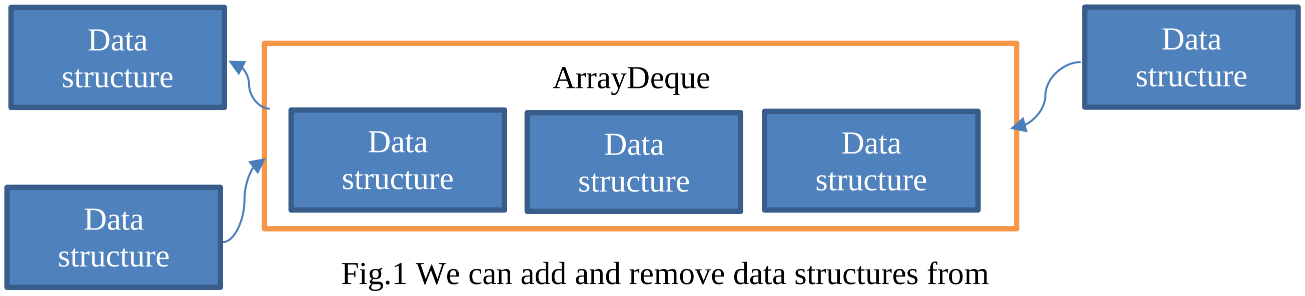

Fig.1 We can add and remove data structures from either end quickly to grow or shrink the Blocked List as needed.

Part 2 of 2 – SLList [Exercises 3.3 and 3.4 in the textbook]

A "Singly-Linked List" is the most basic form of linked list that you will have encountered, and pairs each element of the collection with a single pointer or reference to the subsequent element. For this question you must implement an **SLList** class by completing the implementation provided in the zip file. The implementation should have the following functionality and performance characteristics:

a) **get(i), set(i, x), add(i, x),** and **remove(i)** operations, each running in **O(1 + i)** time.

b) a **reverse()** operation that reverses the order of the **SLList**. This must run in **O(n)** time, must not create any new nodes, and must not use recursion or any type of extra working memory.