American University of Armenia, CSE
CS121 Data Structures
Spring 2024

Homework Assignment 1

Due Date: Friday, February 23 by 23:59 electronically on Moodle

Solve the programming tasks using Java, following good coding practices and required format.

1. (Salamancas | 20 points) Welcome back, special agent Peña. Your days as a DEA officer are long gone, but we still need you on this special case. Today, you find yourself in room 303 of the lovely nursing home, Casa Tranquila. You must communicate with a past resident, Hector Salamanca, and understand his involvement with the powerful Juárez cartel. Don't forget that you were chosen for this mission because of your weird telepathic abilities.

Upon the table, you see a hotel desk bell. You ask, "Hector, are you here?". *Ding*– this means yes. Now, you are to use the bell to communicate with Hector by letting him spell his words letter by letter. You start by going over all the letters of the alphabet one by one and wait to see if the bell rings. If it does, that means you have found the correct letter. You know that a word has ended if the bell doesn't ring for any letter of the alphabet. You are guaranteed that the word will contain at least one and at most ten letters.

Write a method that would accept as an argument a Scanner and would read Hector's message using the approach described above. The ring of the bell should be represented with the letter 'r' and any other character should be counted as an absence of a ring. The method should print the final word at the end.

State and justify the worst case and the best case running times of this algorithm for an alphabet of size $k \times k$ (with k being a non-negative integer number).

–And Now for Something Completely Different

You get tired of going over the whole alphabet each time, and you decide to use a modified strategy. You note that Hector never uses the letter 'z', so you stop considering it. You write the alphabet in a 5x5 table as follows:

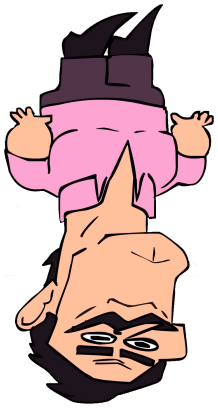
a	b	c	d	e
f	g	h	i	j
k	l	m	n	o
p	q	r	s	t
u	v	w	x	y

You explain to Hector that you are going to start going vertically down from 'a' to determine the row that contains his desired letter (by relying on ringing). Once you locate the row, you start going over the letters of that row, again relying on ringing to find the correct letter. You know that a word has ended if the bell doesn't ring for any row of the table.

Write a method similar to the other one that will communicate with Hector using the approach described above.

State and justify the worst case and the best case running times of this algorithm for an alphabet of size $k \times k$ (with k being a non-negative integer number).

–IT'S LALO TIME [1]



Wait, what happened? You unexpectedly teleported to another location and appeared in an underground methamphetamine laboratory. Someone wants to speak to you. Name's Eduardo Salamanca, but you can call him Lalo. The man got a good head for numbers, so instead of talking to you using a bell, he is going to answer each of your questions with a number– -1, 0 or 1. Moreover, he is also capable of telling you the length of the word in advance (at the very beginning of your communication).

Write a method that, similarly to the other ones, will communicate with Lalo. Come up with an *efficient* approach for doing this.

State and justify the worst case and the best case running times of this algorithm for an alphabet of size $k \times k$ (with k being a non-negative integer number).

2. (YeahMagnets | 15 points) You have a box with n columns ($1 \leq n \leq 100000$) each containing k_i ($1 \leq k_i \leq 200$) metallic cubes put on top of each other. If you hold a magnet next to a side of the box, the cubes will be pulled towards that side as shown in Figure 1. You are given input values

- n– the number of columns of the box,
- n numbers k_i representing the initial number of cubes in each column respectively.

Write an efficient program that outputs the number of cubes in each column after holding a magnet at the **left** side of the box. What is the runtime of this method? Justify.

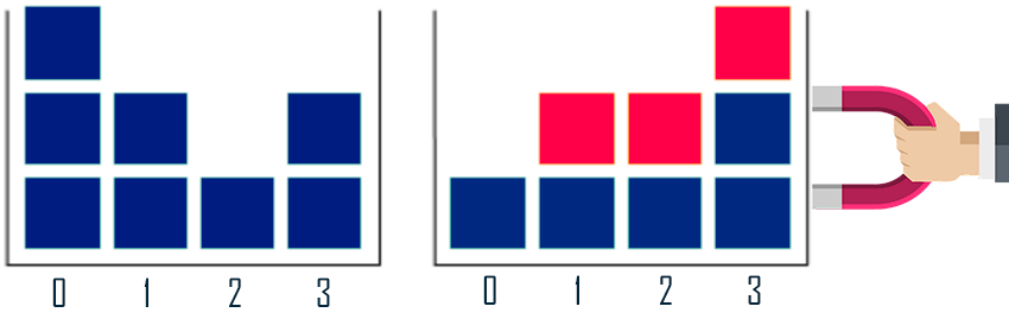


Figure 1: Left: initial Box. Right: the same box after putting a magnet from the right

3. (LinkedMergeSort | 35 points) The task needs you to implement *merge-sort* for a singly-linked structure of unique integer values.

(a) Copy the `Node` class from the `SinglyLinkedList` into this one.

(b) Implement a generic method, that given the first node of a singly-linked structure, prints its contents.

(c) Implement a generic method that, given an array of generic values, creates a singly-linked structure containing the values and returns the first node of the sequence.

(d) Implement a generic method that, given the head node of a singly-linked structure L, splits it into two singly-linked sequences L_{odd} and L_{even} and returns the head nodes of L_{odd} and L_{even}

[1] Contains spoilers!!!! `https://youtu.be/5RJRqtVNfMg?si=9ZUXauptaJPJ60VG`

in a length-2 array. All the elements at odd positions in the original list need to go into L_{odd} and, similarly, all the elements at even positions in the original list need to go into L_{even}.

Your method may traverse the original sequence **only once**. You are **not allowed** to create any nodes or list objects. While we refer to lists L, L_{odd} and L_{even} here, the method should **not** use any list objects; only node sequences.

(e) Implement a method that, given the head nodes of two sorted singly-linked structures of integer elements merges them together into one sorted singly-linked structure and returns its head node.

Your method may traverse the original sequences **only once**. You are **not allowed** to create any nodes or list objects.

(f) Implement a generic method that, given the head node of a singly-linked structure of integer elements applies merge sort on it and returns the head node of the resulting sequence.

(g) Add a main method, and test all the methods above.

4. (StackSort | 15 points) Write a program that sorts a `Stack` of *unique* `Integer`s using *in-place Selection Sort*. The largest element should be at the top, and the smallest should be at the bottom of the stack. You are not allowed to use any additional containers or create any new objects. Your program may include additional helper methods. You are NOT allowed to use `java.util.Stack` functionality. Note that you don't need to swap elements.

5. (QueueDeque | 15 points) Implement the `Deque ADT` using a single `Queue` as the underlying container. Your class should implement the `Deque` interface given in the textbook. You are NOT allowed to use `java.util.Queue` functionality.