Online Assignment 2: Stack (04/12/2021 Saturday)

In your offline component you have already implemented Arr and LL implementation of Stack ADT as specified in the offline specification document. Now your online teachers have forgotten that stack follows a LIFO behaviour. They want to run the following methods using your Arr and LL implementation of Stack ADT:

Fn #	Function	Param.	Ret.	After functions execution	Comment
	[before each function execution.]			<20, 23 , 12, 15>	The values were pushed in the following order: 20, 23, 12, 15. So, TOP points to the value 15. Note that this is a logical definition without any implementation related specification.
7	insert(item, offset)	99, 1		<20, 23 , 99, 12, 15>	Inserts an element, item at a position which is offset (>=0) elements away from the last element (i.e. at TOP – offset according to the logical definition of TOP above). Note that push(x) is not identical to insert(x,0). Invalid values of offset must be handled gracefully.
8	remove(offset)	1	12	<20, 23 , 15>	Remove and return the element from a position which is offset (>=0) elements away from the last element (i.e. at TOP – offset according to the logical definition of TOP above). Invalid values of offset must be handled gracefully.

Now you have tried to humbly argue repeatedly that these are list-like behaviours, but your teachers are not convinced. So, you have to implement these using the Arr and LL implementations of Stack ADT (i.e., your offline assignment). You are only allowed to use your implementations of stack. Your genius

little brother was using one of the PCs in the lab and witnessed the whole drama. Before leaving the lab, he gave you a hint: “Did the teacher put any limit on how many stacks you may use?”

Submission Guidelines:

1. **Create a directory with your 7-digit student id as its name**
2. **You need to create separate files for the Arr implementation code (e.g. Arr1.cpp/Arr1.py), LL implementation code (e.g. LL1.cpp/LL1.py) putting common codes in another file. You will also include your previous implementations as separate files. Create a separate file for the main function to demonstrate the two implementations. YOU cannot use any DS apart from your offline implementations.**
3. **Put all the source files only into the directory created in step 1. Also create a readme.txt file briefly explaining the main purpose of the source files.**
4. **Zip the directory (compress in .zip format. Any other format like .rar, .7z etc. is not acceptable)**
5. **Upload the .zip file on Moodle in the designated assignment submission link. For example, if your student id is 1905xxx, create a directory named 1905xxx. Put only your source files (.c, .cpp, .java, .h, etc.) into 1905xxx. Compress the directory 1905xxx into 1905xxx.zip and upload the 1905xxx.zip on Moodle.**

Failure to follow the above-mentioned submission guideline may result in upto 10% penalty.