Final exam fall 2020
In all the assignments you can look at the **"__main__" == __name__** part of the given code as well as the text files ***expected_out.txt*** for clarification of how the program should behave.

<u>This is a home exam, so all data can be used but it is forbidden to use help from any other people, fellow students or not, or to ask specific questions on forums during the exam.</u>

Good luck!

Multiple choice 20%
These questions are in a separate quiz assignment on Canvas.

(15%) Problem 1: Array operations

You are given the class ArrayList. Implement the operation **reverse_list**. Limitations from previous array assignments apply. In short the only operation you can use directly on the array is the brackets (***[integer_value]***) with a single integer value. Keep in mind you are not allowed to move data to another data structure and back to the list in this problem.

reverse_list(self)
- Reverses the order of the list. So that it is printed before and after the operation is called it is should be reversed.
    - *All other functions in the class must be functional after the reverse operation is called*

(20%) Problem 2: Recursion in SLL

Implement the following operations using singly linked lists. You must use recursion for full marks.

(10%) is_asc_desc_ordered(head)
- Returns true if the list is sorted in either ascending or descending order
    - *both 1,2,3 and 3,2,1 return true but not 1,3,2*
- Returns false otherwise

(10%) count_ascending_series(head)
- Returns the number of ascending series in the list
    - *An ascending series in the list is a series of numbers that grow (or is the same), if the next number shrinks it is not in the series and a new one begins(see example)*
        - 1,2,3,2,3,4,2,7,8 -> 1,2,3|2,3,4|2,7,8 -> 3 series
        - 5,4,3,2,1 -> 5|4|3|2|1 -> 5 series
        - 1,1,1,2,1 -> 1,1,1,2|1 -> 2 series

(25%) Problem 3: Web history with DLL

Write a program that mimics the behaviour of the “back” and “forward” buttons on modern day web browsers(see image). Do this using the structure of a doubly linked list with a current pointer where the current pointer represent the web page currently open.

(15%) nav_to_website(self, web_url)

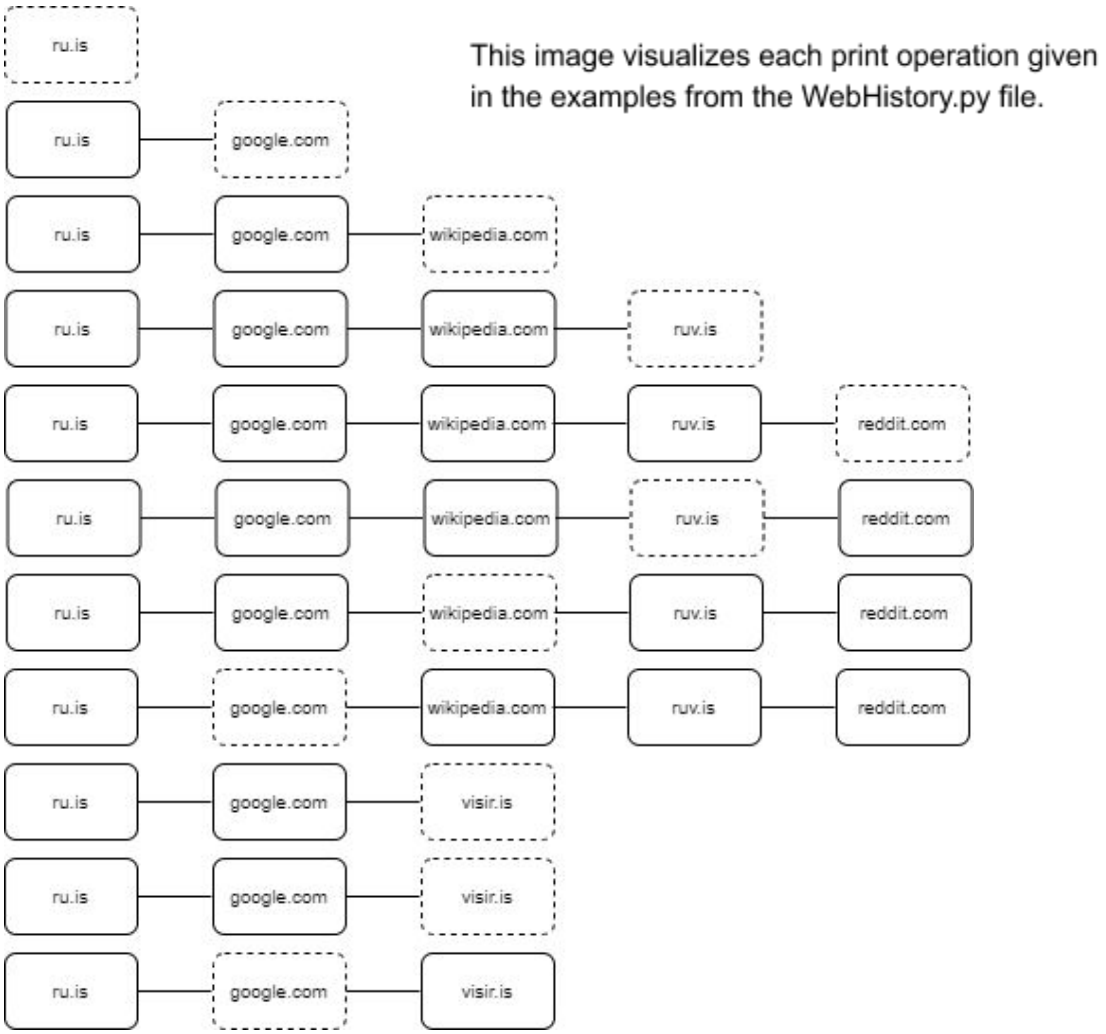
- *Opens the current web_url,* Making it the curr_page
- This also deletes every previously stored forward webpage

(5%) press_back(self)

- Moves the previous pointer back one position (towards the head)
- Do nothing if current page is the page closes to head

(5%) press_forward(self)

- Moves the curr_page pointer forward one position (towards the tail)
- Do nothing if curr_page is the page closes to tail



(20%) Problem 4: BST operations
Implement the **contains_value(self, value)** operation in the BST as well as modifying the **insert()** method so that binary_nodes know the size of the subtree where they are the root.

(10%) contains_value(value)

- Returns **true** if the parameter value exists in the BST
    - *It is recommended to use helper functions for this part of the problem*

(10%) Modify insert to set correct subtree sizes in binary_nodes

- Each node knows how many nodes are in its subtree
    - *A subtree is a specific node, its children, its childrens children, etc.*
    - *Leafs have a size of 1. They are the root of their own subtree with no children.*
- Keep in mind what happens when we call **insert()** for a value that already exists

The image below shows how the tree in the tests is set up and how its each nodes size variable should be set.
V: stands for value
S: stands for size