

Lab 8 Problems

Problem 1 Write a recursive function `first_matches_last(slist)` that takes as an argument a list of strings `slist` and returns a list consisting of those elements of `slist` whose first and last characters are the same. For example,

```
first_matches_last(["abba", "nope", "kook", "hello", "bob"])
```

returns the list `['abba', 'kook', 'bob']`.

Step 1: Write the function in your IDE or in the space below:

Problem 2 Write a function `display(slist)` that takes as an argument `slist`, a list of strings of equal length, and prints the strings in `slist`, one per line, with a border on the top and bottom that is the same size as the strings in `slist`. The border is a string of hyphen characters, '-'. For example, the call

```
display([" * ", " *** ", " ***** ", "*****"])
```

will print the following:

```
-----
 *
 ***
 *****
*****
-----
```

The function `display()` should print the top border, then call a *recursive* helper function to print the elements of the list one per line, and then print the bottom border. The `display()` function may use the string concatenation operator `*`. If `slist` is empty, nothing is printed.

Step 1: Write the function and helper function in your IDE or in the space below:

Problem 3 Suppose we want to write a function to repeat the characters of a string `my_str` to create a new string of a given length. For example, let's say we want to create strings of length 7. Here are some examples of strings and their expansions to strings of length 7:

```
"ab"      -> "abababa"
"cat"     -> "catcatc"
"boat"    -> "boatboa"
```

We can do this using the string multiplication operator `*`, but we need to figure out two things first.

- What do we multiply the string by?
- What if the resulting string is too short?

Let's consider the case of expanding `"ab"` to be of length 7. If we use integer division to determine the multiplier, we get $7 // 2 = 3$. Then `"ab" * 3` is the string `"ababab"`. At that point, we are one character short.

How do we determine how many characters short the result string is, or will be, for any string and any required size? **Hint:** use the modulus operator `%`.

Once we know how many characters "short" the result string is, we can use slicing to get those characters from `my_str` and concatenate them on the end of the result string.

Step 1: Write a function `expand(my_str, size)` that takes a string `my_str` and returns a new string of length `size`, repeating the characters in `my_str` as needed. You can assume that the length of string `my_str` will always be less than or equal to `size`.

Problem 4 In this problem, you will be working with the `BinaryTree` class and the function `insert()` that inserts a new value into a binary search tree.

Step 1: Draw the binary search tree that is created when the following sequence of values are added to a binary search tree: 8, 10, 5, 20, 4, 9.

Step 2: Get the starter code `lab8_starter.py` from the class website (use the Labs link).

Step 3: Add the code to `main()` to create the tree above. Print the tree after each call to `insert()`.