National University of Singapore
School of Computing
CS1101S: Programming Methodology
Semester I, 2024/2025

S6-in-class
Problem Solving and List Processing

LISTS Functions of Source §2

The LISTS functions are a set of predeclared functions provided in Source §2 for list processing. Please refer to the online reference for details of these LISTS functions.

Problems:

1. **[In-class only]** The task for this question will be given by your Avenger during the Studio session.

   Write a function called `remove_duplicates` that takes in a list as its only argument and returns a list with duplicate elements removed. The order of the elements in the returned list does not matter. **Use `accumulate` in your function.**

   ```
   function remove_duplicates(lst) {
       ...
   }
   ```

   Example calls:

   ```
   remove_duplicates(list(1, 2, 3, 4, 4, 3, 2, 1, 2));
   // Result: list(1, 2, 3, 4)

   remove_duplicates(list("a", "x", "b", "c", "c", "b", "d"));
   // Result: list("a", "x", "b", "c", "d")
   ```

2. **[In-class only]** In this question, lists represent *sets*. Each element of the set appears exactly once in its list representation, and the order does not matter. So the list `list(1, 2, 3)` represents the same set as the list `list(3, 2, 1)`.

   In this question, you are supposed to compute all subsets of a give set. Your function `subsets` takes as argument a list, representing the given set, and needs to return a list of lists, each representing a unique subset of the given set.

   ```
   function subsets(xs) {
       ...
   }
   ```

   Example call:

```
subsets(list(1, 2, 3));
// Result: list(list(),
//              list(1), list(2), list(3),
//              list(1,2), list(1,3), list(2,3),
//              list(1,2,3))
```

3. **[In-class only]** A *permutation* of a list s is a list with the same elements as s, but in a possibly different order. For example, the list `list(3,1,2)` is a permutation of `list(1,2,3)`. Write a function `permutations` that takes a list s as argument and returns a list of all permutations of s.

   ```
   function permutations(s) {
       ...
   }
   ```

   Example call:

   ```
   permutations(list(1, 2, 3));
   // Result: list(list(1,2,3), list(1,3,2),
   //              list(2,1,3), list(2,3,1),
   //              list(3,1,2), list(3,2,1))
   ```