

# EECS3311 Software Design

## Winter 2020

### Contracts Exercise

CHEN-WEI WANG

Once you have completed the current version where `imp` is an array, duplicate the starter project and change it to `imp: LINKED_LIST[STRING]`. Then complete the alternative version so you can also practice the use of a linked list.

## 1 Linear Container

A *linear container* is an ordered collection of items with unbounded capacity. Each item or element in a container can be accessed using its absolute position in the container. A position is expressed through an integer index, **starting from 0 and ending at  $count - 1$** , where *count* is the number of items currently stored in the container. For the purpose of this exercise, you are required to implement the following features of a linear container using an array (i.e., a private attribute `imp`). Notice that in this lab exercise, you must implement the `count` as an attribute, not as a query.

- `make`

Initialize an empty container.

- `valid_index (i: INTEGER): BOOLEAN`

Determine if an integer *i* represents a valid index of the current container.

It is also expected that no elements in the container will be changed.

- `get_at (i: INTEGER): STRING`

Given that *i* is a valid index, return the element stored at index *i*.

It is also expected that no elements in the container will be changed.

- `circular_shift_to_left`

Shift each item in the container to the left *circularly* by one position.

For example, given a container `<alan, mark, tom>`, calling `circular_shift_to_left` will change it to `<mark, tom, alan>`. The expected benefits of calling this feature are: **1)** the number of elements is unchanged; and **2)** items in the new container somehow correspond to the old container in the expected way (e.g., `mark` at position 0 in the new container corresponds to `mark` at position 1 in the old container).

- `insert_first (s: STRING)`

Insert value *s* to the first position of the container, under the precondition the input string *s* is not empty.

For example, given a container `<alan, mark, tom>`, calling `insert_first (jim)` will change it to `<jim, alan, mark, tom>`. The expected benefits of calling this feature are: **1)** the number of elements is incremented; **2)** the element stored at the first index now (i.e., 0) is `jim`; and **3)** all indices to the right of the new first index (i.e., indices 1, 2, and 3) have their stored elements (i.e., `alan`, `mark`, and `tom`) somehow corresponding to the old container.

## 2 Your Tasks

- Unzip the starter project `container.zip`

```
cd
unzip container.zip
```

- Open a terminal and type the following command to launch EStudio

Notice that you must type the **&**:

```
cd
estudio19.05 &
```

– Once entering EStudio:

- \* Click on **Add Project...**
  - \* Browse to the unzipped project directory on your home directory.
  - \* Go into the subdirectory **book**, then choose **book.ecf**.
  - \* Click on **Ok**, then **Open**.
- Fill in the implementations and contracts.
  - Restrict yourself to **75 minutes**.
  - Write your own tests to ensure the correctness of the implementation and contracts.

## 3 Receiving Feedback

- Solution will not be made available.
- Speak to your section instructor to go over your solution if you wish.