Assignment #1, Part #1 of 2
Sequential List Implementation

Instructor: Homeyra Pourmohammadali
BME 122 - Data Structures and Algorithms
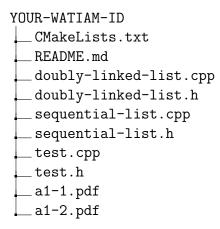Winter 2022 (Online)
UNIVERSITY OF WATERLOO

Due: 5:00 PM, Friday, Jan 21, 2022

Purpose of this assignment

In this assignment, you will practice your knowledge about **array** by implementing a data type called sequential list. The header file `sequential-list.h`, which is explained below, provides the structure of the `SequentialList` class with declarations of member functions. Do not modify the signatures of any of these functions. You need to implement all of the public member functions listed in `sequential-list.cpp`. Note that we set a maximum capacity for this list. That is, when the capacity is reached, the list cannot take any more items.

Instruction

Sign in to GitLab and verify that you have a project set up for your Assignment 1 (A1) at `https://git.uwaterloo.ca/bme122-1221/a1/WATIAM_ID` with the following files.

```
YOUR-WATIAM-ID
├── CMakeLists.txt
├── README.md
├── doubly-linked-list.cpp
├── doubly-linked-list.h
├── sequential-list.cpp
├── sequential-list.h
├── test.cpp
├── test.h
├── a1-1.pdf
└── a1-2.pdf
```

For this part of assignment, you only need to modify `sequential-list.cpp`. Do not change any part of the header file (`sequential-list.h`). You can design your own test case and code in `test.cpp` and `test.h`. It is optional and we will not grade this file.

You can use the same procedures in Assignment 0 to pull, edit, build, commit, and push your repo.

Description

The details of the header file `sequential-list.h` are as follows:

`DataType` defines the kind of data that the list will contain. Being public, it can be accessed directly as `SequentialList::DataType`.

Member variables:
`data_`: Pointer to the memory of array allocated to store the list data.
`capacity_`: Returns the maximum number of elements the list can hold.
`size_`: Returns the number of elements in the list.

Constructor and Destructor:
`SequentialList(unsigned int cap)`: Creates a new SequentialList with the given number of elements.
`SequentialList()`: Destructor, which frees all dynamically allocated memory.
`SequentialList(const SequentialList& rhs)`: Copy constructor, which copies one list `rhs` to another list. For this assignment, you don't have to implement it. Just leave it blank in private section.
`SequentialList& operator=(const SequentialList& rhs)`: Assignment operator for the operations of the form list1 = list2. For this assignment, you don't have to implement it. Just leave it blank in private section.

Constant member functions:
Constant member functions are those function which do not modify class member variables.
`unsigned int size() const`: Returns the number of elements in the list.
`unsigned int capacity() const`: Returns the maximum number of elements the list can hold.
`bool empty() const`: Returns true if the list is empty, false otherwise.
`bool full() const`: Returns true if the list is at capacity, false otherwise.
`DataType select(unsigned int index) const`: Returns the value at the given index in the list. If index is invalid, return the value of the last element. (When the list is empty, $size_$ is 0, any index for select will be invalid. In this case, you can arbitrarily return a data item such as the one with index 0 to satisfy the return data requirement. Note that this is undefined behaviour, so we don't know what this data value will be. It is the left over after previous operations at that memory space. In practice, we will need to check this special case before calling the select method. In this assignment, we will not use any test case related to this special case.)
`unsigned int search(DataType val) const`: Searches for the given value, following the order from head to tail of the list, and returns the index of this value if found (for the first time the value is found). Returns the size of the list if no such value can be found in the list.
`void print() const`: Prints all elements in the list to the standard output. You can choose the printing format, and this will not affect grading.

Non-constant member functions:

These functions can potentially modify member variable of the class.

`bool insert(DataType val, unsigned int index)`: Inserts a value into the list at a given index. Returns true if successful and false otherwise.

`bool insert_front(DataType val)`: Inserts a value at the beginning of the list. Returns true if successful and false otherwise.

`bool insert_back(DataType val)`: Inserts a value at the end of the list. Returns true if successful and false otherwise.

`bool remove(unsigned int index)`: Deletes the value from the list at the given index. Returns true if successful and false otherwise.

`bool remove_front()`: Deletes the value from the beginning of the list. Returns true if successful and false otherwise.

`bool remove_back()`: Deletes the value at the end of the list. Returns true if successful and false otherwise.

`bool replace(unsigned int index, DataType val)`: Replaces the value at the given index with the given value. It is considered as an invalid input when no existing data has been stored at the input index, and in this case it should return false and do nothing. Returns true if successful and false otherwise.

Note: All indexes must start with 0.

Marking

We will try different inputs and check your output. We will only test your program with syntactically and semantically correct inputs.

Part 1 counts 50% of Assignment 1, which is 50 points in total.

Your program runs and does not crash during the test: + 20
Passes Test Cases: + 3 each, in total of 30