Lab 5: Queue and its Applications

Objectives

- To understand the concept of the queue data structure and its implementation using array and singly linked list.
- To write simple applications that use queue and apply more complex operations.

Review

A queue is another special type of list, where the elements are inserted only at the rear of the list and are deleted only at the front of the list. Queue is often called a First-In first-out (FIFO) structure.

Basic Operations

void enqueue(element) adds an element to the tail of the queue.
T dequeue() remove and returns the element at the head of the queue
T queuefront() get the front node of a queue
T queuerear() get the rear node of a queue

Advance Operations

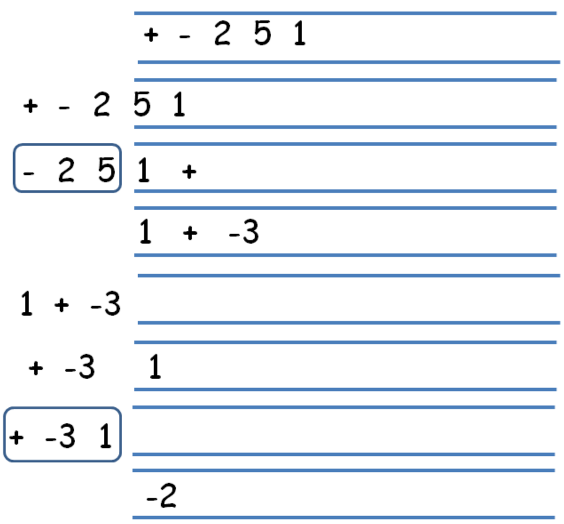
Queue copyQueue() copy content of the current queue to another queue, and return the copied queue at the end
boolean IsIdentical(Queue<T> Q2) which checks whether or not the current queue and the input Queue Q2 are having the same contents in the same order. It returns true if they are the same or return false otherwise. At the end this queue and Q2 should remain the same.

Queue Application

A prefix expression is an expression that an operator is listed before the left and right operands.
A queue data structure can be used to solve the prefix evaluation problem. Following steps are repeated until there is one single element in the queue. Assume first that all items in the prefix expression are loaded to the queue.

```
Inspect the front element.
    If it is an operator followed by all its arguments,
        append the value to the rear of the queue AND THEN
        remove operator and arguments from the front of the queue
    If it is an operator not followed by all its arguments,
        remove it from the front, copy it to the rear.
  it is an operand
        remove it from the front, copy it to the rear.
```

Example of steps for evaluating a prefix expression + - 2 5 1



Round Robin Scheduler

- A round robin scheduler is used to fairly allocate a resource that must be shared by a collection of clients.
- It gives each process a unit of time (time slice, quantum), then move to next process, continue until all processes completed

Exercise 1

a) Implement enqueue(element). Test the result in the main() method. Use the given template source code to complete this exercise.

b) Implement dequeue(). Test the result in the main() method. Use the given template source code to complete this exercise.

c) Implement queuefront()and queuerear(). Test the result in the main() method. Use the given template source code to complete this exercise. You may use the following pseudo code for the implementation.

Exercise 2 Complete the method copyQueue that copies this queue (itself) to another Queue. The copied Queue is returned. Note that the original Queue should remain the same after this method is called.

Exercise 3 Complete the method IsIdentical(Queue<T> Q2) which checks whether or not the queue(itself) and the input Queue Q2 are having the same content and order in the same sequence. At the end this queue and Q2 should remain the same.

Exercise 4 To save you time the following methods are provided.

The method boolean isPrefix(String x, String y, String z) returns true if x is not a number (meaning that it must be an operator) and y and z are integers.

Example

isPrefix(+, 3, 4)	return true
isPrefix(2, 3, 4)	return false
isPrefix(1, +, 2)	return false

Complete the PrefixEval in the template.

The method `String evalPrefixString(String opt, String x, String y)` returns a numerical result of x opt y.

Complete the method PrefixEval

Exercise 5. Implement a round robin scheduler illustration using a queue. The method makeRoundRobin takes 4 input arguments: Q, P, limit, and the resourceAmt. The Q is a queue containing resource demands of people, P is a queue containing names of the persons in the queue, limit is an integer that sets the maximum amount of resource can be given to a person each time, and resourceAmt is an initial amount of resource. The program shows the amount of resource and Queue content after a person gets the amount that he/she needs.