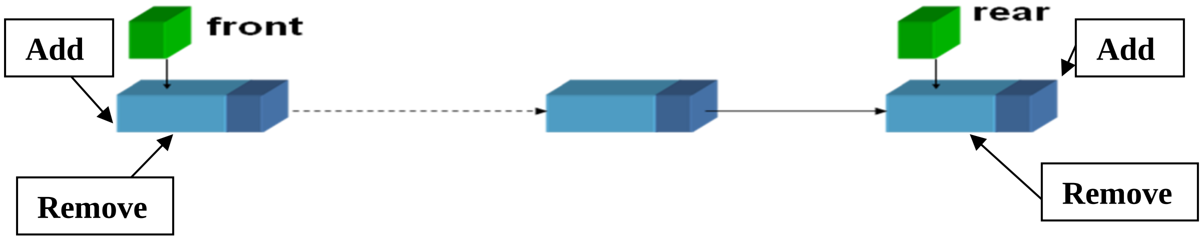
CSCE 2211 Spring 2024 Applied Data Structures
Assignment #2

Dr. Amr Goneid ***Date: Thu Feb 15, Due: Mon Feb 26, 2024***

Part 1: The *DEQ* ADT (50 points)

We recall that the ***Stack*** and ***Queue*** ADT's are sequential containers. For the ***Stack***, both adding and removing an element occur at the same end (Top), while for the ***Queue***, adding an element occurs at one end (Rear) and removal occurs at the other end (Front). As an ADT, the Double-Ended Queue (***DEQ***) is also a sequential container that may function like a Queue or a Stack at both ends. Therefore, it can be used either as a Stack or as a Queue.



The ***DEQ*** ADT can be implemented using a Simple Linked List (***SLL***) with the following member functions:

- **Costructor**: Construct an empty DEQ
- **Destructor**: Destroy DEQ
- **DEQisEmpty**: Test if DEQ is empty
- **Add_Front**: Add an element at the front
- **Add_Rear**: Add an element at the rear
- **Remove_Front**: Remove the element at the front
- **Remove_Rear**: Remove the element at the rear
- **View_Front**: Retrieve the front element without removal
- **View_Rear**: Retrieve the rear element without removal
- **DEQ_Length**: Number of elements in the DEQ

Required Implementation:

Design and implement a template class ***DEQ*** using a SLL with a minimum of the above member functions.

Part 2: Simulation of a Waiting Queue of Planes in an Airport (50 points)

An airport has **one** landing runway. When planes arrive near the airport, they will have to join **one queue**. A plane arriving near the airport **at a random** time $T_{arrival}$, will be instructed to join the queue and it might have to wait (remain airborne) in that landing queue a time T_{wait} until the runway becomes free and ready to receive it.
Once a plane lands on the beginning of the runway, that runway becomes occupied for a fixed time T_s until the plane docks (This is the service time).

Use the ***DEQ*** template class you implemented in Part 1 above to develop a program to simulate the airport queue operations with the objective of computing the average wait time in the landing queue.
Assume the following:

- The time t (clock) unit is one minute.
- A fixed simulation period T_{max}
- A fixed time T_s to complete landing (this is the service time).
- A random arrival time $T_{arrival}$ with a fixed average inter-arrival time ΔT
- No plane will leave the queue until it lands.

You might start your simulation using a “standard run” with:
T_{max} = 6 hours, T_s = 10 minutes, ΔT = 6 minutes

After that, you might investigate the effect of varying arrival rates to simulate prime and slack times of the day, or if the amount of time to complete landing is changed.
Allow your program to produce a “log” of the events of arrival and landing in each run (see methodology of simulating a waiting queue in the file CSCE 2211 Queue Simulation).

Note on using the C++ Random Number Generator (RNG):

Many C++ programs use random numbers generated by a Random Number Generator (RNG). The RNG in C++ is a function **rand( )** that returns a random integer from 0 to 32,767 with equal probability.
To obtain random floating point numbers $0 \leq R < 1.0$ with equal probability, use
float R = rand( ) / float(32767);

To obtain random integers from 1 through n, use
int r = rand( ) % n + 1.

Generally, you may implement a function ***RandInt (i, j)*** that generates an integer between *i* and *j* with equal probability. This is implemented simply as follows:

```
int RandInt (int i , int j)
{
        return rand( ) % (j-i+1) + i ;
}
```

To obtain a random sequence you need to first initialise the RNG using the time of the machine as a seed. This is done so that we do not get the same sequence every time we run the program. The following is an **example** of how to generate a random sequence of pairs of integers with values between 1 and n:

```
#include <time.h>
int x , y, n ;
…………….
srand ( (unsigned) time (NULL) );          //Initialize RNG
repeat as needed                           // Loop over the sequence
{
        x = RandInt (1 , n);               // Generate 1st number
        y = RandInt (1 , n);               // Generate 2nd number
        ………………..                        // Do something with x and y
}
```

One Method to build your own RNG

You may use a random congruence method to code your own RNG. Here is one algorithm to generate a random sequence of Large Integers:
With x_0 = some large int value representing the seed, then a random large integer sequence is obtained as:

$$x_{i+1} = (\alpha\, x_i + \beta) \,\%\, m$$

$\alpha = 25173 \quad \beta = 13849 \quad m = 65536$
The values of x will be between 0 and 65535. You may divide it by m to obtain a random sequence of $0 \leq R < 1.0$ with equal probability.