Data Structures & Algorithms

Prepared by: Mohamed Ayman

Machine Learning Researcher

spring 2019

sw.eng.MohamedAyman@gmail.com

facebook.com/sw.eng.MohamedAyman

linkedin.com/in/eng-mohamed-ayman

codeforces.com/profile/Mohamed_Ayman

Queue

Agenda

1- Queue Definition

2- Queue Representation

3- Basic Operations

4- En-queue (Push) Operation

5- De-queue (Pop) Operation

6- Front Method

7- Empty Method

Let's
STARTUP

Agenda

1- Queue Definition

2- Queue Representation

3- Basic Operations

4- En-queue (Push) Operation

5- De-queue (Pop) Operation

6- Front Method

7- Empty Method

Queue Definition

- Queues are an essential data structure that are found in vast amounts of software from user mode to kernel mode applications that are core to the system.

- Fundamentally they honour a First in First out (FIFO) strategy, that is the item First put into the queue will be the First served, the second item added to the queue will be the second to be served and so on.

- A traditional queue only allows you to access the item at the front of the queue, when you add an item to the queue that item is placed at the back of the queue.

Agenda

✓ 1- Queue Definition

2- Queue Representation

3- Basic Operations

4- En-queue (Push) Operation

5- De-queue (Pop) Operation

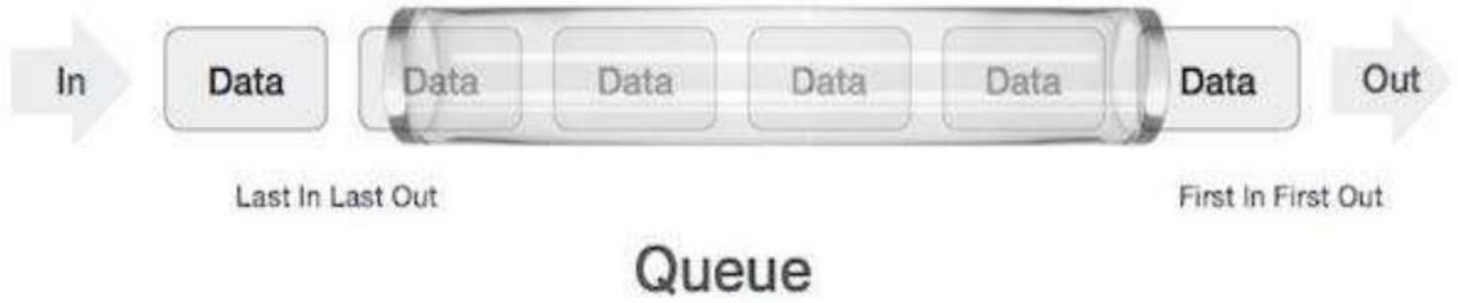
6- Front Method

7- Empty Method

Queue Representation

- As we now understand that in queue, we access both ends for different reasons.

The following diagram given below tries to explain queue representation as data structure.



Queue Node in C++

Link: repl.it/repls/SuperficialAstonishingTelevisions

```
// A queue node
struct node {
    int data;
    node *next;
};

node* head;
```

Agenda

- ✓ 1- Queue Definition
- ✓ 2- Queue Representation
- 3- Basic Operations
- 4- En-queue (Push) Operation
- 5- De-queue (Pop) Operation
- 6- Front Method
- 7- Empty Method

Basic Operations

- Queue operations may involve initializing or defining the queue, utilizing it, and then completely erasing it from the memory.

- Here we shall try to understand the basic operations associated with queues

1- push: add (store) an item to the queue.

2- pop: remove (access) an item from the queue.

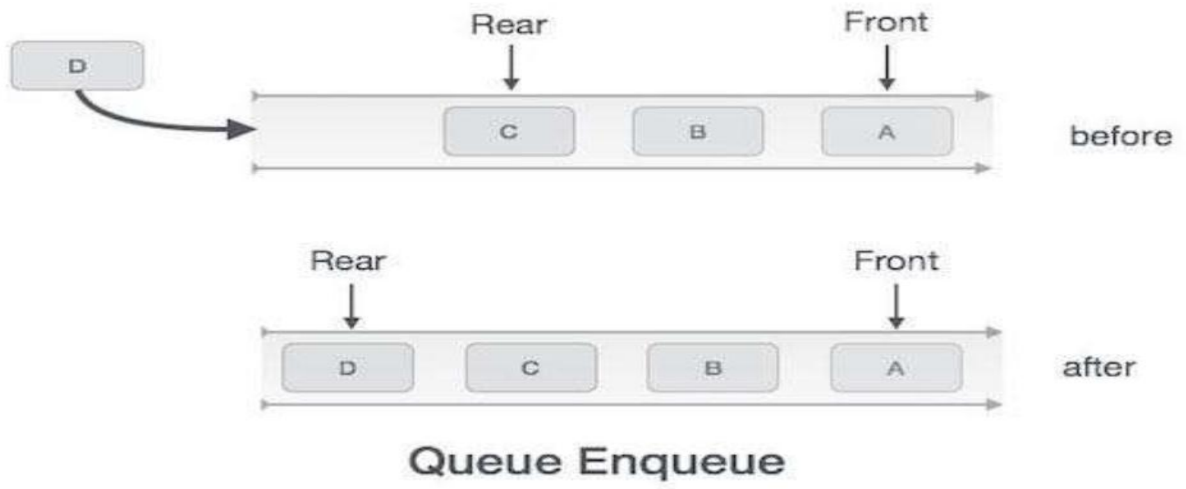
Agenda

- ✓ 1- Queue Definition
- ✓ 2- Queue Representation
- ✓ 3- Basic Operations
- 4- En-queue (Push) Operation
- 5- De-queue (Pop) Operation
- 6- Front Method
- 7- Empty Method

En-queue (Push) Operation

- Queues maintain two data pointers, front and back. Therefore, its operations are comparatively difficult to implement than that of stacks.

- The following steps should be taken to enqueue (insert) data into a queue:



Queue Enqueue

En-queue (Push) Operation

Link: repl.it/repls/SuperficialAstonishingTelevisions

```cpp
// This function add node at end of queue
void push(int new_data) { // O(n)
    // allocate new node and put it's data
    node* new_node = new node();
    new_node->data = new_data;
    // check if the queue is empty
    if(head == NULL) {
        head = new_node;
        return;
    }
    // get last node in queue
    node* curr_node = head;
    while(curr_node->next != NULL)
        curr_node = curr_node->next;
    // make the next of last node as a newNode
    curr_node->next = new_node;
}
```

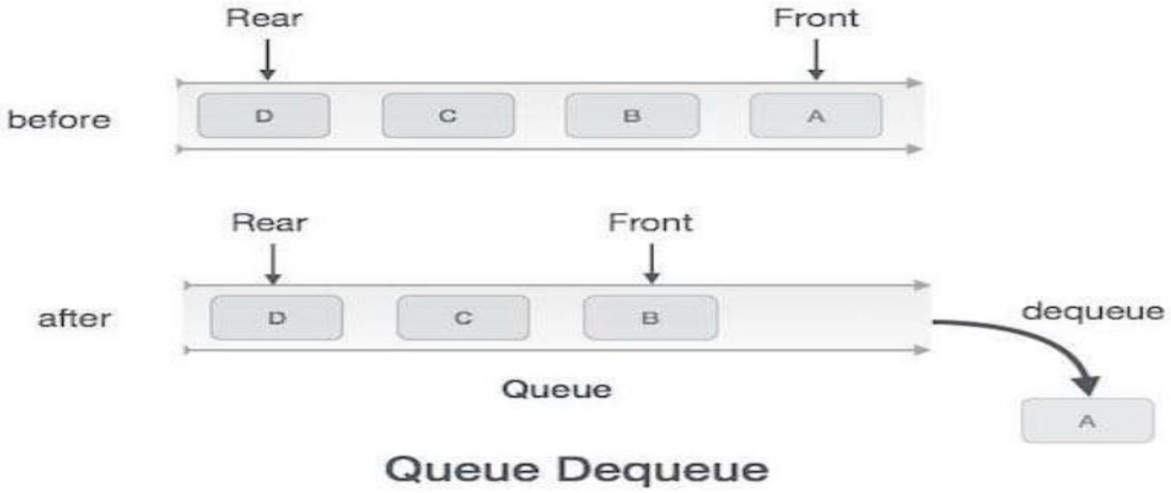
Agenda

- ✓ 1- Queue Definition
- ✓ 2- Queue Representation
- ✓ 3- Basic Operations
- ✓ 4- En-queue (Push) Operation
- 5- De-queue (Pop) Operation
- 6- Front Method
- 7- Empty Method

De-queue (Pop) Operation

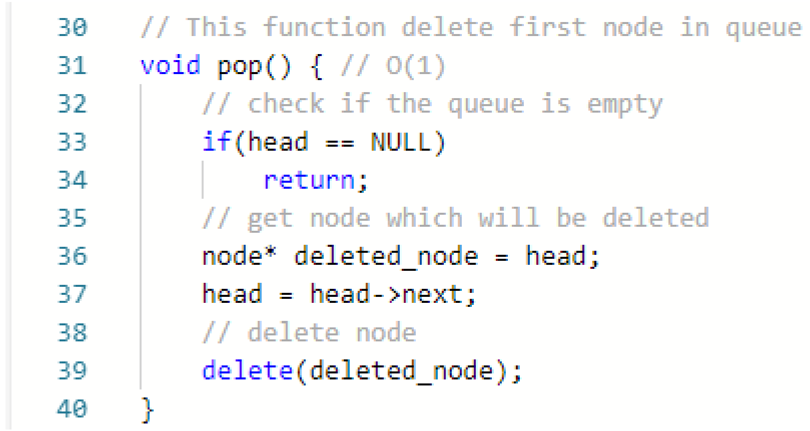
- Accessing data from the queue is a process of two tasks access the data where front is pointing and remove the data after access.

- The following steps are taken to perform dequeue operation:



De-queue (Pop) Operation

Link: repl.it/repls/SuperficialAstonishingTelevisions



```cpp
// This function delete first node in queue
void pop() { // O(1)
    // check if the queue is empty
    if(head == NULL)
        return;
    // get node which will be deleted
    node* deleted_node = head;
    head = head->next;
    // delete node
    delete(deleted_node);
}
```

Agenda

✓ 1- Queue Definition

✓ 2- Queue Representation

✓ 3- Basic Operations

✓ 4- En-queue (Push) Operation

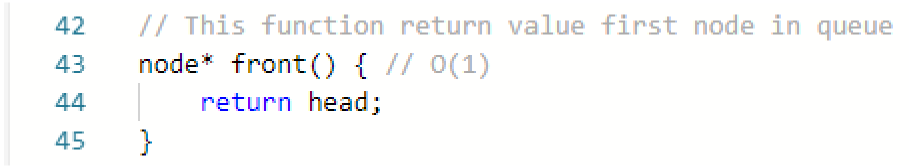
✓ 5- De-queue (Pop) Operation

6- Front Method

7- Empty Method

Front Method

Link: repl.it/repls/SuperficialAstonishingTelevisions



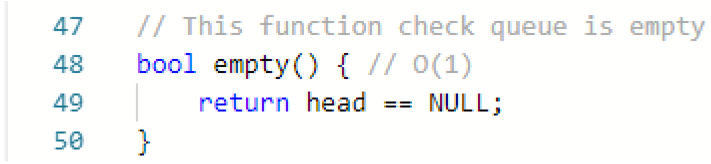
```
// This function return value first node in queue
node* front() { // O(1)
    return head;
}
```

Agenda

- ✓ 1- Queue Definition
- ✓ 2- Queue Representation
- ✓ 3- Basic Operations
- ✓ 4- En-queue (Push) Operation
- ✓ 5- De-queue (Pop) Operation
- ✓ 6- Front Method
- 7- Empty Method

Empty Method

Link: repl.it/repls/SuperficialAstonishingTelevisions



```
// This function check queue is empty
bool empty() { // O(1)
    return head == NULL;
}
```

Agenda

- ✓ 1- Queue Definition
- ✓ 2- Queue Representation
- ✓ 3- Basic Operations
- ✓ 4- En-queue (Push) Operation
- ✓ 5- De-queue (Pop) Operation
- ✓ 6- Front Method
- ✓ 7- Empty Method

DO MORE.

Practice

Practice

1- Sliding Window Maximum summation of all sub-arrays of size k

2- Check string is palindrome or not

3- Generate Binary Numbers from 1 to n

4- Reversing a queue using recursion

5- Reversing the first K elements of a Queue

6- Find the largest multiple of 3

7- Smallest multiple of a given number made of digits 0 and 9 only

8- Minimum time required to rot all oranges

9- Sum of minimum and maximum elements of all subarrays of size k

10- First negative integer in every window of size k

Let's
START UP

Practice

1- Sliding Window Maximum summation of all sub-arrays of size k

2- Check string is palindrome or not

3- Generate Binary Numbers from 1 to n

4- Reversing a queue using recursion

5- Reversing the first K elements of a Queue

6- Find the largest multiple of 3

7- Smallest multiple of a given number made of digits 0 and 9 only

8- Minimum time required to rot all oranges

9- Sum of minimum and maximum elements of all subarrays of size k

10- First negative integer in every window of size k

Sliding Window Maximum summation of all sub-arrays of size k

- Implement function which calculate maximum summation of all sub-arrays of size k and print result
- Function Name: K max_sum
- Parameters: (arr, n, k) arr => array of int which we will process on it

  n => array of int length

  k => window length
- Return: int, max sum of length K in array
- Test Cases: arr = [12, 1, 78, 90, 57, 89, 56], k = 3 => 236

Sliding Window Maximum summation of all sub-arrays of size k

Link: repl.it/repls/ViolentBoilingDemos

```
int K_max_sum(int* arr, int n, int k) {
    // Create a  Queue, Qi that will store each k elements of array
    queue<int>  q;
    int result = 0 , sum = 0;
    // Process first k (or first window) elements of array
    for (int i = 0; i < k; i++) {
        sum += arr[i];
        q.push(arr[i]);
    }
    // try to get maximum sum
    result = max(result, sum);
    // Process rest of the elements, i.e., from arr[k] to arr[n-1]
    for (int i = k ; i < n; i++) {
        // The element at the front of the queue will pop to push new element
        sum -= q.front();
        q.pop();
        sum += arr[i];
        q.push(arr[i]);
        // try to get maximum sum
        result = max(result,sum);
    }
    // Print the maximum element of last window
    return result;
}
```

Check string is palindrome or not

- Implement function which check if string is palindrome or not using queue
- Function Name: is palindrome
- Parameters: str => string which we will process on it
- Return: boolean, True if string is palindrome, False otherwise
- Test Cases: abcdedcba => palindrome

  abcdef => not palindrome

  klmnnmlk => palindrome

Check string is palindrome or not

Link: repl.it/repls/WrongGloomyProcess

```cpp
// A queue based function to check a string is palindrome
bool is_palindrome(string str) {
    queue<char> q;
    // Push all characters of string to queue
    for (int i = 0; i < str.size(); i++)
        q.push(str[i]);
    // Pop all characters of string at queue
    // and compare them with characters of string
    for (int i = str.size()-1; i >= 0; i--) {
        if(q.front() != str[i])
            return false;
        q.pop();
    }
    return true;
}
```

Generate Binary Numbers from 1 to n

- Implement function which generate numbers from 1 to n in binary representation
- Function Name: generate binary
- Parameters: (n)   n => limit of binary numbers to print it
- Return: None
- Test Cases: n = 2   =>   1, 10

  n = 5   =>   1, 10, 11, 100, 101

Generate Binary Numbers from 1 to n

Following is an interesting method that uses queue data structure to print binary numbers.

1) Create an empty queue of strings

2) Enqueue the first binary number "1" to queue.

3) Now run a loop for generating and printing n binary numbers.

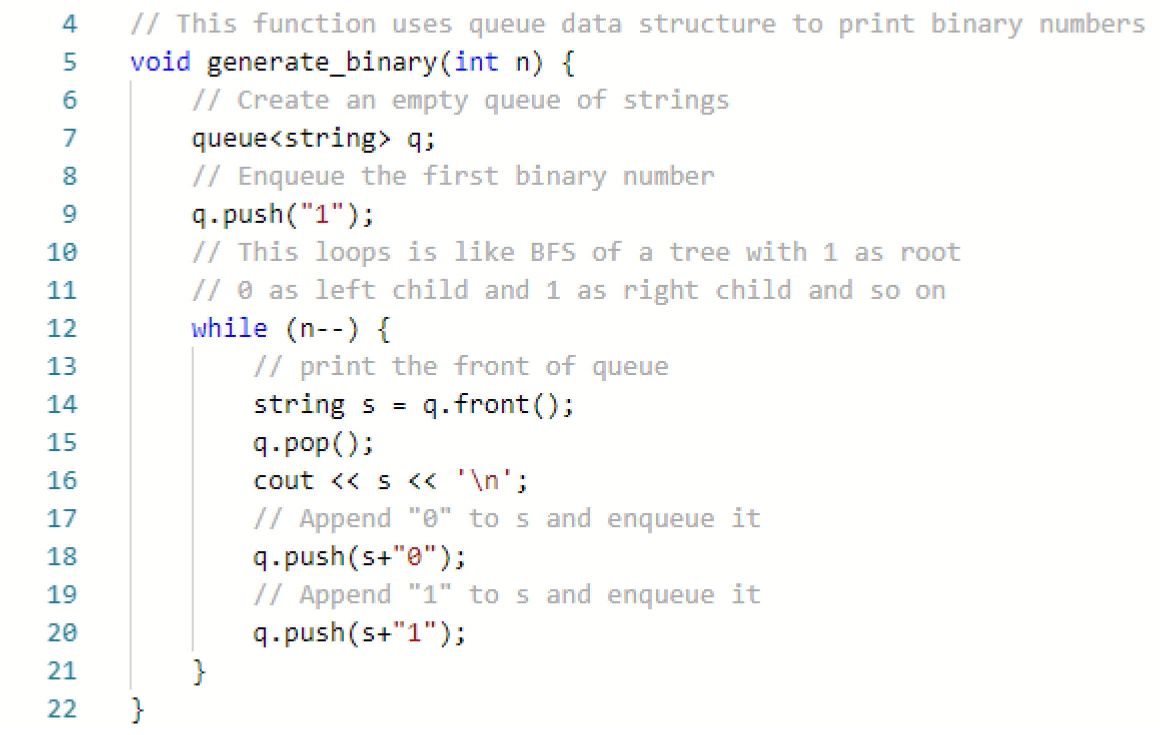
……a) Dequeue and Print the front of queue.

……b) Append "0" at the end of front item and enqueue it.

……c) Append "1" at the end of front item and enqueue it.

Generate Binary Numbers from 1 to n

Link: repl.it/repls/NovelAwkwardLivecd



```cpp
// This function uses queue data structure to print binary numbers
void generate_binary(int n) {
    // Create an empty queue of strings
    queue<string> q;
    // Enqueue the first binary number
    q.push("1");
    // This loops is like BFS of a tree with 1 as root
    // 0 as left child and 1 as right child and so on
    while (n--) {
        // print the front of queue
        string s = q.front();
        q.pop();
        cout << s << '\n';
        // Append "0" to s and enqueue it
        q.push(s+"0");
        // Append "1" to s and enqueue it
        q.push(s+"1");
    }
}
```

Reversing a queue using recursion

- Implement function which reverse queue using recursion
- Function Name: reverse queue
- Parameters: (q)   q => queue which we will process on it
- Return: None
- Test Cases:

  [56, 27, 30, 45, 85, 92, 58, 80, 90, 100] =>

  100 90 80 58 92 85 45 30 27 56

Reversing a queue using recursion

Recursive Algorithm :

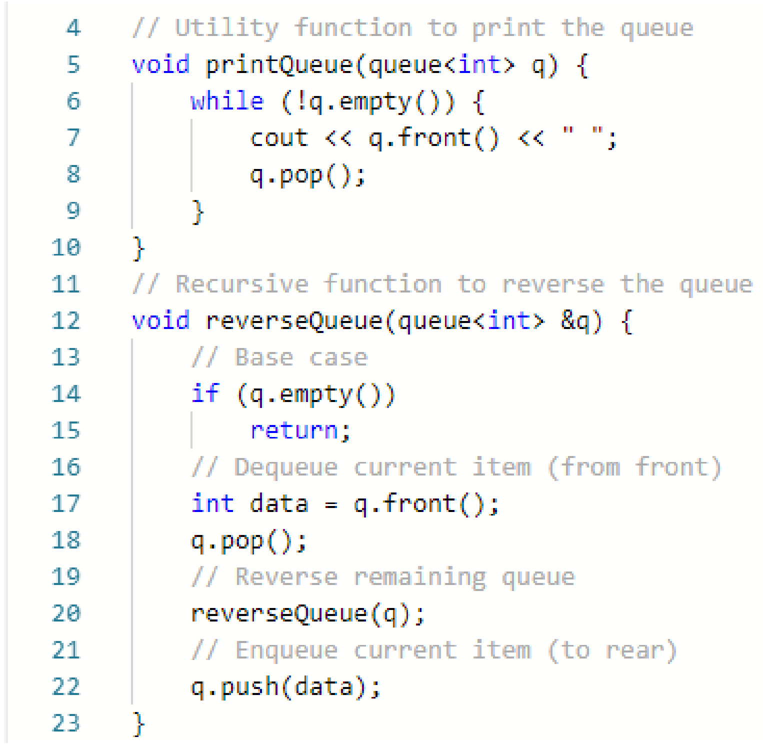
1) Pop element from the queue if the queue has elements otherwise return empty queue.
2) Call reverseQueue function for the remaining queue.
3) Push the popped element in the resultant reversed queue.

Pseudo Code :

```
queue reverseFunction(queue)
{
    if (queue is empty)
       return queue;
    else {
       data = queue.front()
       queue.pop()
       queue = reverseFunction(queue);
       q.push(data);
       return queue;
    }
}
```

Reversing a queue using recursion

Link: repl.it/repls/CulturedSuddenArraylist



```cpp
// Utility function to print the queue
void printQueue(queue<int> q) {
    while (!q.empty()) {
        cout << q.front() << " ";
        q.pop();
    }
}
// Recursive function to reverse the queue
void reverseQueue(queue<int> &q) {
    // Base case
    if (q.empty())
        return;
    // Dequeue current item (from front)
    int data = q.front();
    q.pop();
    // Reverse remaining queue
    reverseQueue(q);
    // Enqueue current item (to rear)
    q.push(data);
}
```

Reversing the first K elements of a Queue

- Implement function which reverse the K elements in queue
- Function Name: reverse queue first K elements
- Parameters: (k, q) q => queue which we will process on it

  k => the first k elements
- Return: None
- Test Cases:

  [56, 27, 30, 45, 85, 92, 58, 80, 90, 100] =>

  85 45 30 27 56 92 58 80 90 100

Reversing the first K elements of a Queue

The idea is to use an auxiliary stack.

1) Create an empty stack.

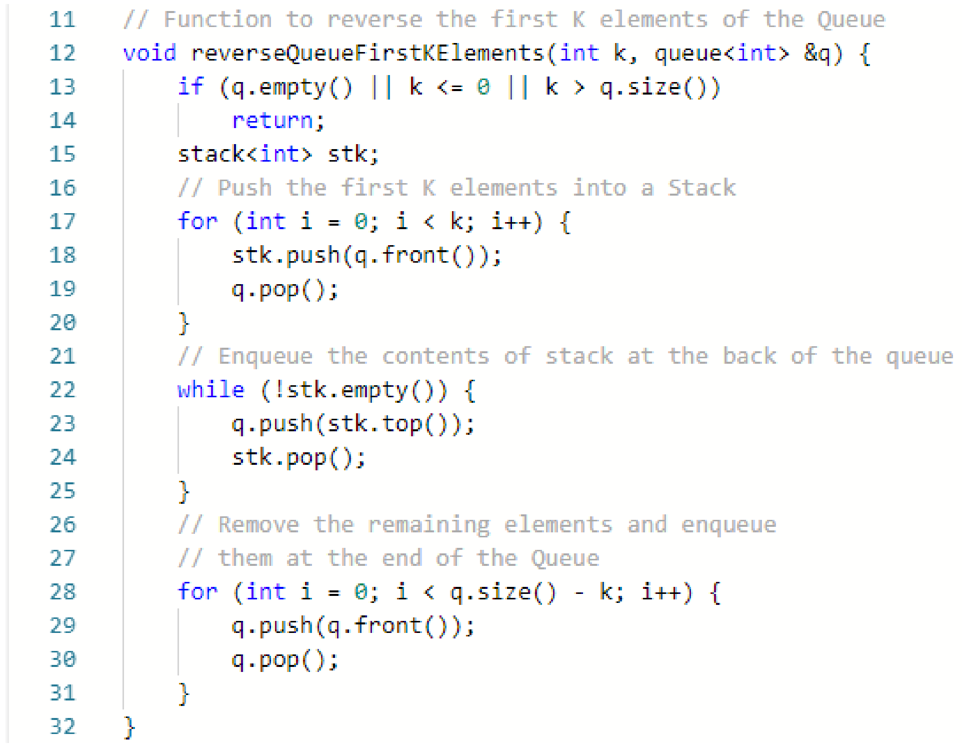
2) One by one dequeue items from given queue and push the dequeued items to stack.

3) Enqueue the contents of stack at the back of the queue

4) Reverse the whole queue.

Reversing the first K elements of a Queue

Link: repl.it/repls/ThunderousAccurateAutocad



```cpp
// Function to reverse the first K elements of the Queue
void reverseQueueFirstKElements(int k, queue<int> &q) {
    if (q.empty() || k <= 0 || k > q.size())
        return;
    stack<int> stk;
    // Push the first K elements into a Stack
    for (int i = 0; i < k; i++) {
        stk.push(q.front());
        q.pop();
    }
    // Enqueue the contents of stack at the back of the queue
    while (!stk.empty()) {
        q.push(stk.top());
        stk.pop();
    }
    // Remove the remaining elements and enqueue
    // them at the end of the Queue
    for (int i = 0; i < q.size() - k; i++) {
        q.push(q.front());
        q.pop();
    }
}
```

Find the largest multiple of 3

Problem Link: geeksforgeeks.org/find-the-largest-number-multiple-of-3

Smallest multiple of a given number made of digits 0 and 9 only

Problem Link: geeksforgeeks.org/smallest-multiple-of-a-given-number-made-of-digits-0-and-9-only

Minimum time required to rot all oranges

Problem Link: geeksforgeeks.org/minimum-time-required-so-that-all-oranges-become-rotten

Sum of minimum and maximum elements of all subarrays of size k

Problem Link: geeksforgeeks.org/sum-minimum-maximum-elements-subarrays-size-k

First negative integer in every window of size k

Problem Link: geeksforgeeks.org/first-negative-integer-every-window-size-k

Practice

1- ~~Sliding Window Maximum summation of all sub-arrays of size k~~

2- ~~Check string is palindrome or not~~

3- ~~Generate Binary Numbers from 1 to n~~

4- ~~Reversing a queue using recursion~~

5- ~~Reversing the first K elements of a Queue~~

6- ~~Find the largest multiple of 3~~

7- ~~Smallest multiple of a given number made of digits 0 and 9 only~~

8- ~~Minimum time required to rot all oranges~~

9- ~~Sum of minimum and maximum elements of all subarrays of size k~~

10- ~~First negative integer in every window of size k~~

DO
MORE.

Assignment

References

[01]	Online Course YouTube Playlists	https://bit.ly/2Pq88rN
[02]	Introduction to Algorithms Thomas H. Cormen	https://bit.ly/2ONhuSn
[03]	Competitive Programming 3 Steven Halim	https://nus.edu/2z4OvyK
[04]	Fundamental of Algorithmics Gilles Brassard and Paul Bartley	https://bit.ly/2QuvwbM
[05]	Analysis of Algorithms An Active Learning Approach	http://bit.ly/2EgCCYX
[06]	Data Structures and Algorithms Annotated Reference	http://bit.ly/2c37XEv
[07]	Competitive Programmer's Handbook	https://bit.ly/2APAbeG
[08]	GeeksforGeeks	https://geeksforgeeks.org
[09]	Codeforces Online Judge	http://codeforces.com
[10]	HackerEarth Online Judge	https://hackerearth.com
[11]	TopCoder Online Judge	https://topcoder.com

Questions ?