Mahros Mohamed

223106831

Assignment: queue;

1. Give an algorithm for reversing a queue Q. To access the queue, we are only allowed to use the methods of queue ADT.

```cpp
// reverse queue
#include <iostream>
#include <queue>
using namespace std;

void reverseQueue(queue<int>& q) {
    if (q.empty()) return;

    int item = q.front();
    q.pop();

    reverseQueue(q);

    q.push(item);
}

int main() {
    queue<int> q;
    q.push(1);
    q.push(2);
    q.push(3);
    q.push(4);
    q.push(5);

    reverseQueue(q);

    while (!q.empty()) {
        cout << q.front() << " ";
        q.pop();
    }
    return 0;
}
```

2. How can you implement a queue using two stacks?

```cpp
// implement a queue using two stacks
#include <iostream>
#include <stack>

class Queue {
    private:
        stack<int> s1, s2;

    public:
        void enqueue(int x) {
            s1.push(x);
        }

        int dequeue() {
            if (s2.empty()) {
                if (s1.empty()) {
                    throw out_of_range("Queue is empty");
                }
                while (!s1.empty()) {
                    s2.push(s1.top());
                    s1.pop();
                }
            }
            int item = s2.top();
            s2.pop();
            return item;
        }

        bool isEmpty() {
            return s1.empty() && s2.empty();
        }
};

int main() {
```

```
    Queue q;
    q.enqueue(1);
    q.enqueue(2);
    q.enqueue(3);

    cout << q.dequeue() << endl; // 1
    cout << q.dequeue() << endl; // 2

    q.enqueue(4);
    cout << q.dequeue() << endl; // 3
    cout << q.dequeue() << endl; // 4

    return 0;
}
```

3. Given a queue of integers, rearrange the elements by interleaving the first half of the list with the second half of the list. For example, suppose a queue stores the following sequence of values: [11, 12, 13, 14, 15, 16, 17, 18, 19, 20]. Consider the two halves of this list: first half: [11, 12, 13, 14, 15] second half: [16, 17, 18, 19, 20]. These are combined in an alternating fashion to form a sequence of interleave pairs: the first values from each half (11 and 16), then the second values from each half (12 and 17), then the third values from each half (13 and 18), and so on. In each pair, the value from the first half appears before the value from the second half. Thus, after the call, the queue stores the following values: [11, 16, 12, 17, 13, 18, 14, 19, 15, 20].

```
// Rearranging a Queue by Interleaving Two Halves


#include <iostream>
#include <queue>
#include <stack>

void interleaveQueue(queue<int>& q) {
    stack<int> s;
    int n = q.size();
    int half = n / 2;

    // Step 1: Push the first half onto the stack
    for (int i = 0; i < half; ++i) {
        s.push(q.front());
        q.pop();
    }

    // Step 2: Enqueue the stack elements back to the queue
    while (!s.empty()) {
        q.push(s.top());
        s.pop();
    }
```

```
    // Step 3: Enqueue the second half back to the queue
    int size = q.size();
    for (int i = 0; i < size - half; ++i) {
        q.push(q.front());
        q.pop();
    }

    // Step 4: Interleave the two halves
    while (!s.empty() || size - half > 0) {
        if (!s.empty()) {
            q.push(s.top());
            s.pop();
        }
        if (size - half > 0) {
            q.push(q.front());
            q.pop();
            --size;
        }
    }
}

int main() {
    queue<int> q;
    for (int i = 1; i <= 10; ++i) {
        q.push(i);
    }

    interleaveQueue(q);

    while (!q.empty()) {
        cout << q.front() << " ";
        q.pop();
    }

    return 0;
}
```

4. Given an integer k and a queue of integers, how do you reverse the order of the first k elements of the queue, leaving the other elements in the same relative order? For example, if k=4 and queue has the elements [10, 20, 30, 40, 50, 60, 70, 80, 90]; the output should be [40, 30, 20, 10, 50, 60, 70, 80, 90].

```
// Reverse the Order of the First k Elements in a Queue
```

```cpp
#include <iostream>
#include <queue>
#include <stack>

void reverseFirstK(queue<int>& q, int k) {
    if (k <= 0 || k > q.size()) return;

    stack<int> s;

    // Step 1: Push the first k elements into the stack
    for (int i = 0; i < k; i++) {
        s.push(q.front());
        q.pop();
    }

    // Step 2: Enqueue the elements back to the queue
    while (!s.empty()) {
        q.push(s.top());
        s.pop();
    }

    // Step 3: Move the remaining elements to the back of the queue
    int size = q.size();
    for (int i = 0; i < size - k; i++) {
        q.push(q.front());
        q.pop();
    }
}

int main() {
    queue<int> q;
    for (int i = 10; i <= 90; i += 10) {
        q.push(i);
    }

    int k = 4;
    reverseFirstK(q, k);

    while (!q.empty()) {
        cout << q.front() << " ";
        q.pop();
    }

    return 0;
}
```

5. Describe the output for the following sequence of queue operations: enqueue(5), enqueue(3), dequeue(), enqueue(2), enqueue(8), dequeue(), dequeue(), enqueue(9), enqueue(1), dequeue(), enqueue(7), enqueue(6), dequeue(), dequeue(), enqueue(4), dequeue(), dequeue().

```
// Output for a Sequence of Queue Operations


#include <iostream>
#include <queue>

void queueOperations() {
    queue<int> q;

    q.push(5);
    q.push(3);
    q.pop(); // Removes 5
    q.push(2);
    q.push(8);
    q.pop(); // Removes 3
    q.pop(); // Removes 2
    q.push(9);
    q.push(1);
    q.pop(); // Removes 8
    q.push(7);
    q.push(6);
    q.pop(); // Removes 9
    q.pop(); // Removes 1
    q.push(4);
    q.pop(); // Removes 7
    q.pop(); // Removes 6

    // Print remaining elements in the queue
    while (!q.empty()) {
        cout << q.front() << " ";
        q.pop();
    }
}

int main() {
    queueOperations(); // Call the function to perform operations
    return 0;
}
```