UCS301 - Lab Assignment 4: Queues (C++ Code Solutions)

By:- Vansh Arora (1024030514)

Q1)

```
#include <iostream>
#define MAX 100
using namespace std;

class Queue {
    int arr[MAX];
    int front, rear;
public:
    Queue() { front = 0; rear = 0; }
    bool isEmpty() { return front == rear; }
    bool isFull() { return rear == MAX; }

    void enqueue(int x) {
        if (isFull()) { cout << "Queue is full\n"; return; }
        arr[rear++] = x;
    }
    int dequeue() {
        if (isEmpty()) { cout << "Queue is empty\n"; return -1; }
        return arr[front++];
    }
    int peek() {
        if (isEmpty()) { cout << "Queue is empty\n"; return -1; }
        return arr[front];
    }
    void display() {
        if (isEmpty()) { cout << "Queue is empty\n"; return; }
        cout << "Queue: ";
        for (int i = front; i < rear; i++) cout << arr[i] << " ";
        cout << endl;
    }
};

int main() {
    Queue q;
    int choice, x;
    while (true) {
        cout << "\n1.Enqueue  2.Dequeue  3.Peek  4.Display  5.IsEmpty  6.IsFull
0.Exit\n";
        cout << "Choice: "; cin >> choice;
        switch(choice) {
            case 1: cout << "Value: "; cin >> x; q.enqueue(x); break;
            case 2: x = q.dequeue(); if (x!=-1) cout << "Dequeued: " << x <<
endl; break;
            case 3: x = q.peek(); if (x!=-1) cout << "Front: " << x << endl;
break;
            case 4: q.display(); break;
            case 5: cout << (q.isEmpty()? "Empty\n":"Not Empty\n"); break;
```

```
            case 6: cout << (q.isFull()? "Full\n":"Not Full\n"); break;
            case 0: return 0;
            default: cout << "Invalid\n";
        }
    }
}
```

Q2)

```
#include <iostream>
#define MAX 5
using namespace std;

class CircularQueue {
    int arr[MAX];
    int front, rear;
public:
    CircularQueue() { front = -1; rear = -1; }
    bool isEmpty() { return front == -1; }
    bool isFull() { return (front == 0 && rear == MAX-1) || (rear == (front-
1)%(MAX-1)); }

    void enqueue(int x) {
        if (isFull()) { cout << "Queue is full\n"; return; }
        if (front == -1) { front = rear = 0; arr[rear] = x; }
        else if (rear == MAX-1 && front != 0) { rear = 0; arr[rear] = x; }
        else arr[++rear] = x;
    }
    int dequeue() {
        if (isEmpty()) { cout << "Queue is empty\n"; return -1; }
        int data = arr[front];
        if (front == rear) front = rear = -1;
        else if (front == MAX-1) front = 0;
        else front++;
        return data;
    }
    int peek() {
        if (isEmpty()) { cout << "Queue is empty\n"; return -1; }
        return arr[front];
    }
    void display() {
        if (isEmpty()) { cout << "Queue is empty\n"; return; }
        cout << "Queue: ";
        if (rear >= front) {
            for (int i = front; i <= rear; i++) cout << arr[i] << " ";
        } else {
            for (int i = front; i < MAX; i++) cout << arr[i] << " ";
            for (int i = 0; i <= rear; i++) cout << arr[i] << " ";
        }
        cout << endl;
    }
};
```

```
int main() {
    CircularQueue q;
    int choice, x;
    while (true) {
        cout << "\n1.Enqueue  2.Dequeue  3.Peek  4.Display  0.Exit\n";
        cout << "Choice: "; cin >> choice;
        switch(choice) {
            case 1: cout << "Value: "; cin >> x; q.enqueue(x); break;
            case 2: x = q.dequeue(); if (x!=-1) cout << "Dequeued: " << x <<
endl; break;
            case 3: x = q.peek(); if (x!=-1) cout << "Front: " << x << endl;
break;
            case 4: q.display(); break;
            case 0: return 0;
            default: cout << "Invalid\n";
        }
    }
}
```

Q3)

```
#include <iostream>
#include <queue>
using namespace std;

void interleaveQueue(queue<int>& q) {
    int n = q.size();
    if (n % 2 != 0) { cout << "Queue size must be even\n"; return; }

    queue<int> firstHalf;
    for (int i = 0; i < n/2; i++) {
        firstHalf.push(q.front()); q.pop();
    }

    while (!firstHalf.empty()) {
        q.push(firstHalf.front()); firstHalf.pop();
        q.push(q.front()); q.pop();
    }
}

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

    interleaveQueue(q);
    cout << "Interleaved Queue: ";
    while (!q.empty()) { cout << q.front() << " "; q.pop(); }
}
```

Q4)

```
#include <iostream>
```

```
#include <queue>
#include <unordered_map>
using namespace std;

char firstNonRepeating(string s) {
    unordered_map<char,int> freq;
    queue<char> q;
    for (char c : s) {
        freq[c]++;
        q.push(c);
        while (!q.empty() && freq[q.front()] > 1) q.pop();
    }
    if (!q.empty()) return q.front();
    return '#';
}

int main() {
    string s; cout << "Enter string: "; cin >> s;
    char res = firstNonRepeating(s);
    if (res != '#') cout << "First non-repeating: " << res;
    else cout << "No non-repeating character";
}
```

Q5a)

```
#include <iostream>
#include <queue>
using namespace std;

class Stack {
    queue<int> q1, q2;
public:
    void push(int x) {
        q2.push(x);
        while (!q1.empty()) { q2.push(q1.front()); q1.pop(); }
        swap(q1,q2);
    }
    int pop() {
        if (q1.empty()) return -1;
        int x = q1.front(); q1.pop();
        return x;
    }
    int top() { return q1.empty()? -1 : q1.front(); }
    bool empty() { return q1.empty(); }
};

int main() {
    Stack s;
    s.push(10); s.push(20); s.push(30);
    cout << s.pop() << endl;
    cout << s.top() << endl;
}
```

Q5b)

```
#include <iostream>
#include <queue>
using namespace std;

class Stack {
    queue<int> q;
public:
    void push(int x) {
        q.push(x);
        for (int i = 0; i < q.size()-1; i++) {
            q.push(q.front()); q.pop();
        }
    }
    int pop() {
        if (q.empty()) return -1;
        int x = q.front(); q.pop();
        return x;
    }
    int top() { return q.empty()? -1 : q.front(); }
    bool empty() { return q.empty(); }
};

int main() {
    Stack s;
    s.push(10); s.push(20); s.push(30);
    cout << s.pop() << endl;
    cout << s.top() << endl;
}
```

Additional Q1)

```
#include <iostream>
#include <queue>
using namespace std;

void generateBinary(int n) {
    queue<string> q;
    q.push("1");
    for (int i=0; i<n; i++) {
        string s = q.front(); q.pop();
        cout << s << " ";
        q.push(s + "0");
        q.push(s + "1");
    }
}

int main() {
    int n; cout << "Enter n: "; cin >> n;
    generateBinary(n);
}
```

Additional Q2)

```
#include <iostream>
#include <queue>
using namespace std;

void insert(queue<int>& q, int val) {
    if (q.empty() || val <= q.front()) {
        q.push(val);
        return;
    }
    int temp = q.front(); q.pop();
    insert(q, val);
    q.push(temp);
}

void sortQueue(queue<int>& q) {
    if (q.empty()) return;
    int x = q.front(); q.pop();
    sortQueue(q);
    insert(q, x);
}

int main() {
    queue<int> q;
    q.push(3); q.push(1); q.push(4); q.push(2);
    sortQueue(q);
    while (!q.empty()) { cout << q.front() << " "; q.pop(); }
}
```

Additional Q3)

```
#include <iostream>
#include <queue>
#include <stack>
using namespace std;

bool checkSorted(queue<int> q) {
    stack<int> st;
    int expected = 1, n = q.size();

    while (!q.empty()) {
        int front = q.front(); q.pop();
        if (front == expected) expected++;
        else {
            if (!st.empty() && st.top() < front) return false;
            st.push(front);
        }
        while (!st.empty() && st.top() == expected) {
            st.pop(); expected++;
        }
    }
    return expected == n+1;
```

```
}

int main() {
    queue<int> q;
    q.push(5); q.push(1); q.push(2); q.push(3); q.push(4);
    cout << (checkSorted(q)? "Yes":"No");
}
```

Additional Q4)

```
#include <iostream>
#include <queue>
using namespace std;

int countStudents(vector<int>& students, vector<int>& sandwiches) {
    queue<int> q;
    for (int s : students) q.push(s);
    int index = 0, count = 0;
    while (!q.empty() && count < q.size()) {
        if (q.front() == sandwiches[index]) {
            q.pop(); index++; count = 0;
        } else {
            q.push(q.front()); q.pop();
            count++;
        }
    }
    return q.size();
}

int main() {
    vector<int> students = {1,1,0,0};
    vector<int> sandwiches = {0,1,0,1};
    cout << countStudents(students, sandwiches);
}
```