CS221-L Data Structures and Algorithms Lab

Lab report # 06

Submitted by: Shayan Rizwan

Registration Number: 2024585

Submitted to: Adnan Haider

Semester: 3rd

Faculty of Computer Science and Engineering

GIK Institute of Engineering Sciences and Technology

Task 1:

Write a program that takes a queue of integers and an integer K as input and reverses the first K elements of the queue, while keeping the order of the rest of the elements the same.

Example:

Input: Queue = [10, 20, 30, 40, 50], $K = 3$

Output: [30, 20, 10, 40, 50]

Use a stack to reverse the first K elements.

Reinsert the reversed part back into the queue.

Solution

CODE:

```
#include <iostream>

using namespace std;

class Queue {
public:
  int front;
  int rear;
  int arr[100];

  Queue() : front(-1), rear(-1) {}

  bool isEmpty() { return front == -1 || front > rear; }

  bool isFull() { return rear == 100 - 1; }

  int getFront() {
    if (isEmpty()) {
      cout << "Queue is empty" << endl;
      return -1;
    }
    return arr[front];
  }
```

```
int getRear() {
  if (isEmpty()) {
    cout << "Queue is empty" << endl;
    return -1;
  }
  return arr[rear];
}

void enqueue(int val) {
  if (isFull()) {
    cout << "Queue is full" << endl;
    return;
  }
  if (isEmpty())
    front = 0;

  rear++;
  arr[rear] = val;
}

int dequeue() {
  if (isEmpty()) {
    cout << "Queue is empty" << endl;
    return -1;
  }
  int ans = arr[front];
  front++;
  if (isEmpty())
    front = rear = -1;
  return ans;
}

void display() {
  if (isEmpty()) {
    cout << "Queue is empty" << endl;
    return;
  }
  cout << "Queue: ";
  for (int i = front; i <= rear; i++) {
    cout << arr[i] << " ";
  }
  cout << endl;
}
```

```
};

// Simple stack implemented with array
class Stack {
public:
  int top;
  int arr[100];

  Stack() : top(-1) {}

  bool isEmpty() {
    return top == -1;
  }

  bool isFull() {
    return top == 99;
  }

  void push(int val) {
    if (isFull()) {
      cout << "Stack overflow" << endl;
      return;
    }
    top++;
    arr[top] = val;
  }

  int pop() {
    if (isEmpty()) {
      cout << "Stack underflow" << endl;
      return -1;
    }
    int val = arr[top];
    top--;
    return val;
  }
};

// Function to reverse first k elements of the queue using the stack
void reverseFirstK(Queue &q, int k) {
  if (q.isEmpty() || k <= 0) return;

  if (k > (q.rear - q.front + 1)) {
```

```
    cout << "K is greater than the queue size. Reversing entire queue." <<
endl;
    k = q.rear - q.front + 1;
  }

  Stack s;

  // Step 1: Dequeue first k elements and push to stack
  for (int i = 0; i < k; i++) {
    int val = q.dequeue();
    s.push(val);
  }

  // Step 2: Pop from stack and enqueue back to queue (reversed part)
  while (!s.isEmpty()) {
    int val = s.pop();
    q.enqueue(val);
  }

  // Step 3: Move the remaining elements (size-k) to the back of the queue to
maintain order
  int remaining = (q.rear - q.front + 1) - k;
  for (int i = 0; i < remaining; i++) {
    int val = q.dequeue();
    q.enqueue(val);
  }
}




int main() {
  Queue q;

  q.enqueue(10);
  q.enqueue(20);
  q.enqueue(30);
  q.enqueue(40);
  q.enqueue(50);
  q.enqueue(60);

  cout << "Original queue:" << endl;
  q.display();
```

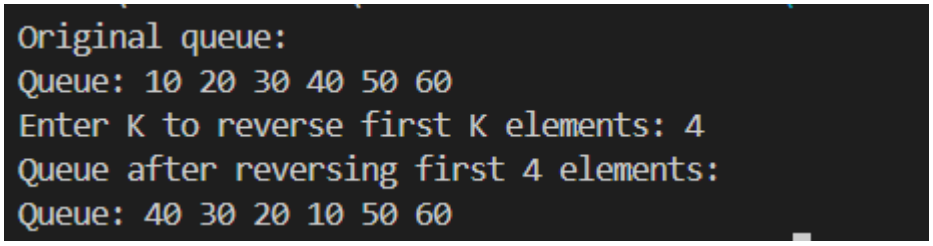
```
  cout << "Enter K to reverse first K elements: ";
  int k;
  cin >> k;

  reverseFirstK(q, k);

  cout << "Queue after reversing first " << k << " elements:" << endl;
  q.display();

  return 0;
}
```

OUTPUT



Task 2:

Given a queue with even number of elements, interleave the first half of the queue with the second half.

Example:

Input: [10, 20, 30, 40, 50, 60]

Output: [10, 40, 20, 50, 30, 60]

Solution

CODE:

```
#include <iostream>

using namespace std;

class Queue {
```

```
public:
  int front;
  int rear;
  int arr[100];

  Queue() : front(-1), rear(-1) {}

  bool isEmpty() { return front == -1 || front > rear; }

  bool isFull() { return rear == 100 - 1; }

  int getFront() {
    if (isEmpty()) {
      cout << "Queue is empty" << endl;
      return -1;
    }
    return arr[front];
  }

  int getRear() {
    if (isEmpty()) {
      cout << "Queue is empty" << endl;
      return -1;
    }
    return arr[rear];
  }

  void enqueue(int val) {
    if (isFull()) {
      cout << "Queue is full" << endl;
      return;
    }
    if (isEmpty())
      front = 0;

    rear++;
    arr[rear] = val;
  }

  int dequeue() {
    if (isEmpty()) {
      cout << "Queue is empty" << endl;
      return -1;
    }
```

```
    int ans = arr[front];
    front++;
    if (isEmpty())
      front = rear = -1;
    return ans;
  }

  void display() {
    if (isEmpty()) {
      cout << "Queue is empty" << endl;
      return;
    }
    cout << "Queue: ";
    for (int i = front; i <= rear; i++) {
      cout << arr[i] << " ";
    }
    cout << endl;
  }
};


void interleaveQueue(Queue &q) {
  if (q.isEmpty()) return;

  int size = q.rear - q.front + 1;

  if (size % 2 != 0) {
    cout << "Queue does not have an even number of elements." << endl;
    return;
  }

  int halfSize = size / 2;

  int temp[100];
  for (int i = 0; i < halfSize; i++) {
    temp[i] = q.dequeue();
  }

  for (int i = 0; i < halfSize; i++) {
    q.enqueue(temp[i]);      // first half element
    q.enqueue(q.dequeue());  // second half element
  }
}
```

```
int main() {

  Queue q2;
  q2.enqueue(10);
  q2.enqueue(20);
  q2.enqueue(30);
  q2.enqueue(40);
  q2.enqueue(50);
  q2.enqueue(60);

  cout << "Original queue:" << endl;
  q2.display();



  // After enqueueing your elements
  interleaveQueue(q2);
  q2.display();



  return 0;
}
```

OUTPUT

```
Original queue:
Queue: 10 20 30 40 50 60
Queue: 10 40 20 50 30 60
```