

- **Orientation**

- **Week 1 Information**

- **Lesson 1: Linear Structures**

- **Week 1 Graded Activities**



- [Quiz: Week 1 Quiz](#)
10 questions



- [Reading: Important Tips and Notes for All Challenge Problems](#)
10 min



- [Reading: Guidelines for Asking for Help With Code](#)
10 min



- [Quiz: Week 1 Challenge](#)
1 question



- [Programming Assignment: Linked Lists and Merge Sort Project](#)
5h



Quiz • 30 min

Week 1 Challenge



Submit your assignment

Due May 25, 12:29 PM IST

Try again



Receive grade

To Pass 80% or higher

Grade

100%

View Feedback

We keep your highest score



Confirm Navigation

Are you sure you want to leave this page?

Stay on this Page

Leave this Page

Confirm Navigation

Are you sure you want to leave this page?

Stay on this Page

Leave this Page



Week 1 Challenge

Graded Quiz • 30 min

Due May 25, 12:29 PM IST



Congratulations! You passed!
To Pass 80% or higher

Keep Learning

Grade
100%

Week 1 Challenge

Latest Submission Grade
100%

1.Question 1

Write functions that reverse the elements in a stack and in a queue. The starter code below include the STL `<stack>` and `<queue>` data structures.

A stack of integers is declared as `"std::stack<int>"` and the stack's `top()` member function returns the integer at the top of the stack (but also leaves it at the top of the stack). The `push()` method pushes a new integer onto the top of the stack and the `pop()` method deletes the value at the top of the stack.

A queue of integers is declared as `"std::queue<int>"` and the queue's `front()` member function returns the integer at the front of the queue (but also leaves it at the front of the queue). The `push()` method pushes a new integer onto the back of the queue and the `pop()` method deletes the value at the front of the queue.

Your job is to implement procedures that reverse the order of elements in a stack, and in a queue. The procedures `print_stack()` and `print_queue()` are provided to help you see if your procedures work.

```
1  #include <iostream>
2  #include <stack>
3  #include <queue>
4  #include<vector>
5
6  std::stack<int> reverse_stack(std::stack<int> s) {
7      std::stack<int> reversed_s;
8
9  }
```

```

10     for(int i=0;i<5;i++)
11     {
12         reversed_s.push(s.top());
13         s.pop();
14     }
15
16     return reversed_s;
17 }
18
19
20 std::queue<int> reverse_queue(std::queue<int> q) {
21     std::queue<int> reversed_q;
22     int s = q.size();
23
24     // Second queue
25
26
27     for (int i = 0; i < s; i++) {
28
29         // Get the last element to the
30         // front of queue
31         for (int j = 0; j < q.size() - 1; j++) {
32             int x = q.front();
33             q.pop();
34             q.push(x);
35         }
36
37         // Get the last element and
38         // add it to the new queue
39         reversed_q.push(q.front());
40         q.pop();
41     }
42
43
44     return reversed_q;
45 }
46
47 void print_stack(std::string name, std::stack<int> s) {
48     std::cout << "stack " << name << ": ";
49     while (!s.empty()) {
50         std::cout << s.top() << " ";
51         s.pop();
52     }
53     std::cout << std::endl;
54 }
55
56 void print_queue(std::string name, std::queue<int> q) {
57     std::cout << "queue " << name << ": ";
58     while (!q.empty()) {
59         std::cout << q.front() << " ";
60         q.pop();
61     }
62     std::cout << std::endl;
63 }
64
65 int main() {

```

```
66  std::stack<int> s, rs;
67  std::queue<int> q, rq;
68
69  s.push(1); s.push(2); s.push(3); s.push(4); s.push(5);
70
71  print_stack("s",s);
72
73  rs = reverse_stack(s);
74
75  print_stack("reversed s",rs);
76
77  q.push(1); q.push(2); q.push(3); q.push(4); q.push(5);
78
79  print_queue("q",q);
80
81  rq = reverse_queue(q);
82
83  print_queue("reversed q",rq);
84
85  return 0;
```

Run Reset



Correct

stack sent: 93 15 77 86 83
stack returned: 83 86 77 15 93
queue sent: 35 86 92 49 21
queue returned: 21 49 92 86 35

5 / 5 points