

## Deque using stack

// Function to push element x to the back of the deque.

```
void push_back_pb(deque<int> &dq, int x) {  
    dq.push_back(x);  
}
```

// Function to pop element from the back of the deque.

```
void pop_back_ppb(deque<int> &dq) {  
    if (!dq.empty())  
        dq.pop_back();  
    else  
        return;  
}
```

// Function to return element from the front of the deque.

```
int front_dq(deque<int> &dq) {  
    if (!dq.empty())  
        return dq.front();  
    else  
        return -1;  
}
```

// Function to push element x to the front of the deque.

```
void push_front_pf(deque<int> &dq, int x) {  
    dq.push_front(x);  
}
```

The screenshot displays a C++ IDE interface. On the left, the 'Output Window' shows 'Compilation Results' for a problem solved successfully. It reports 10/10 test cases passed, 100% accuracy, 2/2 points scored, and a time taken of 0.02. The 'Solve Next' button is visible at the bottom. On the right, the C++ code is shown, implementing a deque using stack operations. The code includes functions for pushing and popping from the back, returning the front element, and pushing to the front. The code is as follows:

```
1 // } Driver Code Ends  
2  
3 // User function Template for C++  
4  
5 // dq : deque in which element is to be pushed  
6 // x : element to be pushed  
7  
8 // Function to push element x to the back of the deque.  
9 void push_back_pb(deque<int> &dq, int x) {  
10     dq.push_back(x);  
11 }  
12  
13 // Function to pop element from the back of the deque.  
14 void pop_back_ppb(deque<int> &dq) {  
15     if (!dq.empty())  
16         dq.pop_back();  
17     else  
18         return;  
19 }  
20  
21 // Function to return element from the front of the deque.  
22 int front_dq(deque<int> &dq) {  
23     if (!dq.empty())  
24         return dq.front();  
25     else  
26         return -1;  
27 }  
28  
29 // Function to push element x to the front of the deque.  
30 void push_front_pf(deque<int> &dq, int x) {  
31     dq.push_front(x);  
32 }
```