4️⃣

Deque

What is Deque?

Double Ended Queue which is also called Deque is a type of queue data structure in which insertion and deletion of elements can be either in front or rear.

Deque in STL.

Syntax:

```
deque<object_type> variable_name;
```

Example:

```
stack<int> s;
stack<string> s;
```

The Whole Code :

```
// Containers --> deque
#include <bits/stdc++.h>
using namespace std;
void explainDeque()
```

```
{
      deque<int> dq;
      dq.push_back(1);  // {1}
      dq.emplace_back(2); //{1,2}
      dq.push_front(4); //{4,1,2}
      dq.emplace_front(3); // {3,4,1,2}

      dq.pop_back(); // {3,4,1}
      dq.pop_front(); // {4,1}

      dq.back();
      dq.front();

      // rest functions same as vector
      // begin, end, rbegin, rend, clear, insert, size , swap
}
int main()
{
      explainDeque();
      return 0;
}
```

Function in Deque —

push_back() – to insert an element at the end of the deque.

```
deque<int> dq;
dq.push_back(110);
dq.push_back(220);
```

push_front() – to insert an element at the front of the deque.

```
deque<int> dq;
dq.push_front(110);
dq.push_front(220);
```

pop_back() – deletes the last element of the deque.

```
dq.pop_back();
```

pop_front() – deletes the front element of the deque.

```
dq.pop_front();
```

front() – it gives a reference to the first element of the deque.

```
dq.front();
```

back() – it gives a reference to the last element of the deque.

```
dq.back();
```

size() – returns the number of elements on the deque.

```
dq.size();
```

empty() – to check if the deque is empty or not.

```
dq.empty();
```

Striver’s Code

```
#include<bits/stdc++.h>
using namespace std;
void printdeque(deque<int> dq)
{
    deque<int>::iterator it;
    for(it=dq.begin();it!=dq.end();it++)
    {
        cout<<*it<<" ";
```

```
    }
    cout<<endl;
}
int main()
{
    deque<int> dq;
    dq.push_back(10);
    dq.push_back(20);
    dq.push_front(30);
    dq.push_front(40);
    dq.push_front(50);

    cout<<"The elements in the deque are: ";
    printdeque(dq);

    cout<<"The size of the deque is: "<<dq.size()<<endl;
    cout<<"The first element in the deque: "<<dq.front()<<endl;
    cout<<"Deleting the first element"<<endl;
    dq.pop_front();
    printdeque(dq);
    cout<<"The last element of the deque: "<<dq.back()<<endl;
    cout<<"Deleting the last element"<<endl;
    dq.pop_back();
    printdeque(dq);

}

Output:

The elements in the deque are: 50 40 30 10 20
The size of the deque is: 5
The first element in the deque: 50
Deleting the first element
40 30 10 20
The last element of the deque: 20
Deleting the last element
40 30 10
```

Other functions in Deque:

- **begin()** – it refers to the first element of the deque.
- **end()** – it refers to the theoretical element after the last element of the deque.
- **cbegin()** – it refers to the first element of the deque.
- **cend()** – it refers to the theoretical element after the last element of the deque
- **rbegin()** – it points to the last element of the deque.
- **rend()** – it points to the theoretical element before the first element of the deque.
- **emplace_front()** – to insert an element at the front of the deque.

- **emplace_back()** – to insert an element at the end of the deque.
- **max_size()** – the maximum elements a deque can hold.
- **clear()** – to delete all the elements of the deque.
- **erase()** – to delete a single element or elements between a particular range.