CIRCULAR LINKED LIST

CODE:

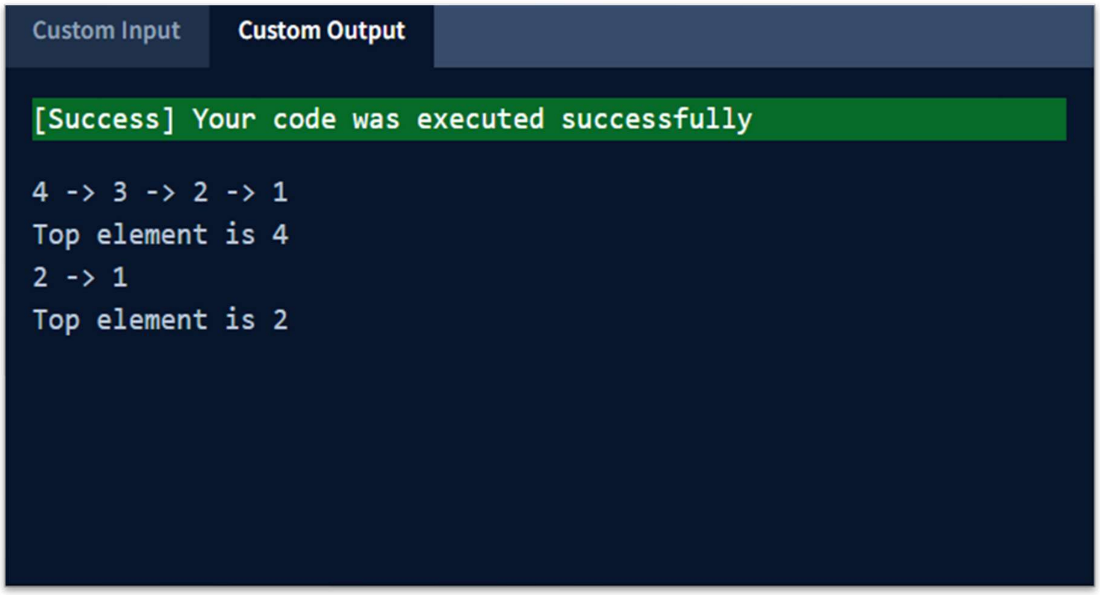
```
#include <iostream>
using namespace std;
class CNode {
private:
    int data;
    CNode* next; // Change "new" to "next"
    friend class CLL;
};
class CLL {
public:
    CLL();
    ~CLL();
    bool empty() const;
    const int front() const;
    const int back() const; // Add back() function declaration
    void advance();
    void add(const int);
    void remove();
private:
    CNode* cursor;
};
CLL::CLL() {
    cursor = new CNode(); // Initialize cursor with a new node
    cursor->next = cursor; // Point to itself to create an empty
circular list
}
CLL::~CLL() {
    while (!empty()) {
        remove();
    }
```

```
}
bool CLL::empty() const {
   return (cursor->next == cursor);
}
const int CLL::front() const {
   return (cursor->next->data);
}
const int CLL::back() const {
   return (cursor->data);
}
void CLL::advance() {
   cursor = cursor->next;
}

void CLL::add(const int num) {
   CNode* newNode = new CNode();
   newNode->data = num;
   newNode->next = cursor->next;
   cursor->next = newNode;
}
void CLL::remove() {
   CNode* old = cursor->next;
   cout << "Deleted node is " << old->data;
   cursor->next = old->next;
   delete (old);
}
int main() {
   CLL myCircularList;
   myCircularList.add(1);
   myCircularList.add(2);
   myCircularList.add(3);
   cout << "Front: " << myCircularList.front() << endl;
   cout << "Back: " << myCircularList.back() << endl;
   myCircularList.remove();
```

```
    cout << "Front after removal: " << myCircularList.front() <<
endl;
    return 0;
}
```

OUTPUT:



```
[Success] Your code was executed successfully

4 -> 3 -> 2 -> 1
Top element is 4
2 -> 1
Top element is 2
```