assgn4_D1171708 Brian

In this assignment. Firstly. we have to finish the Node.cpp with the default constructor. and the data with the constructor.

```cpp
#include "Node.h"
#include <iostream>
Node::Node(){
    elem = 0;
    prev = nullptr;
    next = nullptr;
}

Node::Node(int e){
    elem = e;
    prev = nullptr;
    next = nullptr;
}
```

Next, we have to complete each function due to the IQueue.h file.

```cpp
    IQueue(); // Default constructor.

    void enqueue(int); // Enqueue operation.

    int dequeue(); // Dequeue operation.

    int headElem(); // Check head element of the queue.

    bool isEmpty(); // Check whether the queue is empty.

    int getSize(); // Get the size of the queue.

    Node *getHead(); // Get the head pointer of the queue.

    Node *getTail(); // Get the tail pointer of the queue.

    void printHeadToTail(); // Print the queue from head to tail.
```

The concept of enqueue is adds an element to the rear of the queue. and for dequeue is removes and returns the element from the front of the queue.

```
#include <iostream>
#include <iomanip>
#include "IQueue.h"
using namespace std;

IQueue::IQueue() {
    head = nullptr;
    tail = nullptr;
}

void IQueue::enqueue(int e) {
    Node *newNode = new Node(e); //c++ malloc
    if (isEmpty()) {
        head = newNode;
        tail = newNode;
    } else {
        tail->next = newNode;
        newNode->prev = tail;
        tail = newNode;
    }
}
```

```
int IQueue::dequeue() {
    if (isEmpty()) {
        return 0; // no queue
    } else {
        int value = head->elem;
        Node *temp = head;
        head = head->next;
        if (head != NULL) {
            head->prev = NULL;
        } else {
            tail = NULL;
        }
        delete temp; //  c++ free.
        return value;
    }
}

int IQueue::headElem() {
    if (isEmpty()) {
        return 0;
    } else {
        return head->elem;
    }
}
```

```
bool IQueue::isEmpty() {
    return head == nullptr;
}

int IQueue::getSize() {
    int size = 0;
    Node *current = head;
    while (current != nullptr) {
        size++;
        current = current->next;
    }
    return size;
}

Node *IQueue::getHead() {
    return head;
}

Node *IQueue::getTail() {
    return tail;
}
```

```
void IQueue::printHeadToTail() {
    Node *current = head;
    int count = 0;
    while (current != nullptr) {
        count = count + 1;
        cout << setw(3) <<current->elem <<"   ";
        if(count%20==0){
            cout << endl;
        }
        current = current->next;

    }
    cout <<endl;
}
```

Use setw(3) to align the cout elements from <iomanip> library

```cpp
using namespace std;

int main() {

    int trail;
    int enqueue, dequeue;
    int elem;
    srand(time(NULL));
    trail  = rand() % 10 + 1;
    IQueue queue;
    cout << "Trial count: " << trail <<endl;
    for (int i = 0; i < trail; ++i) {
        enqueue = rand() % 100;
        cout << "\n>>>>Trial " << i + 1 << ": enqueue and dequeue operations"<<endl;
        cout << "Enqueue " <<  enqueue << " elements to the queue."<<endl;
        for (int j = 0; j < enqueue; ++j) {
            elem = rand() % 100;
            queue.enqueue(elem);
        }
        cout << "Current queue size: " << queue.getSize() << ". Content of the queue from head to tail:"<<endl;
        queue.printHeadToTail();

        dequeue = rand() % queue.getSize();
        cout << "\n\nDequeue " <<  dequeue << " elements to the queue."<<endl;
        for (int j = 0; j < dequeue; ++j) {
            queue.dequeue();
        }
        cout << "Current queue size: " << queue.getSize() << ". Content of the queue from head to tail"<<endl;
        queue.printHeadToTail();
        cout << "-----------------------------------------------------------------\n"<< endl;
    }

    return 0;
}
```

Last, use the rand() to generate the random trail and the enqueue dequeue elements and print it all out.