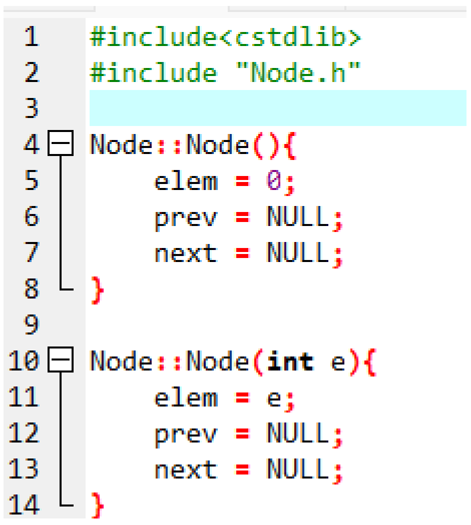
Report of assignment4

Jensen D1265209

Node.cpp



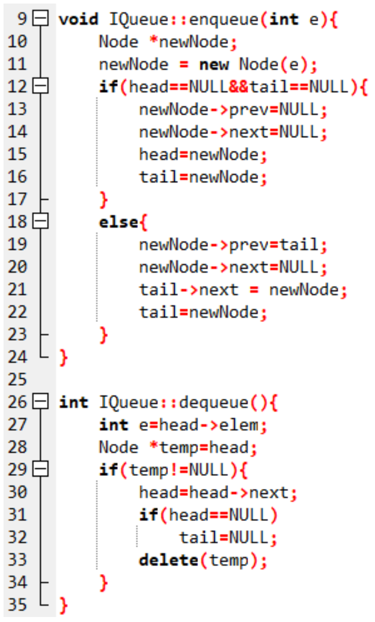
```cpp
#include<cstdlib>
#include "Node.h"

Node::Node(){
    elem = 0;
    prev = NULL;
    next = NULL;
}

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

This part is to initialize the element in side Node, including elem, prev, and next. There are two different types of Node, one is elem equal to 0 in the beginning, and the other is equal to one number. Apart from this, prev and next equal to NULL in the beginning.

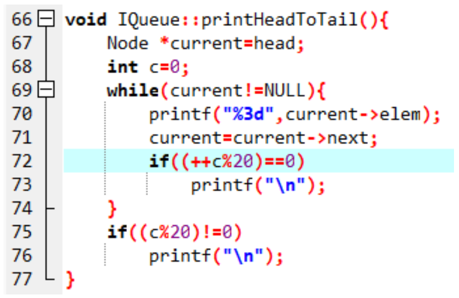
IQueue.cpp



```cpp
void IQueue::enqueue(int e){
    Node *newNode;
    newNode = new Node(e);
    if(head==NULL&&tail==NULL){
        newNode->prev=NULL;
        newNode->next=NULL;
        head=newNode;
        tail=newNode;
    }
    else{
        newNode->prev=tail;
        newNode->next=NULL;
        tail->next = newNode;
        tail=newNode;
    }
}

int IQueue::dequeue(){
    int e=head->elem;
    Node *temp=head;
    if(temp!=NULL){
        head=head->next;
        if(head==NULL)
            tail=NULL;
        delete(temp);
    }
}
```

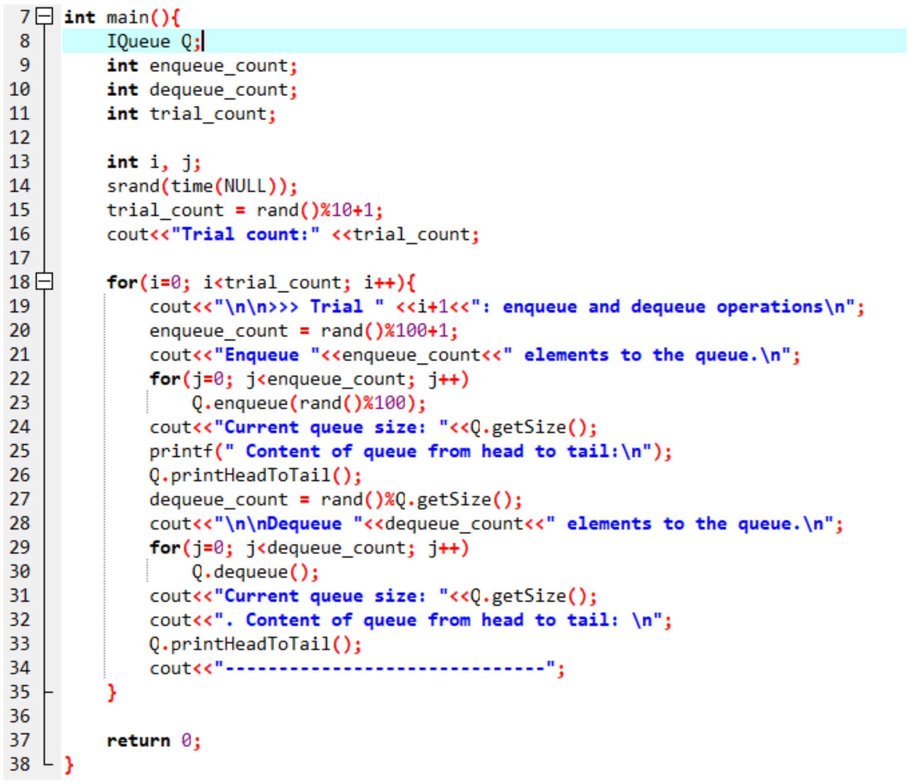
This part is similar with the Liner Link in c, however there are only few situations need to consider in this assignment. First, enqueue, one case is no element in side queue, another case is there are elements in side queue and need to add new elements from the end of queue. Second, dequeue, need to delete the elements from the head. And clean the element one by one.



```cpp
void IQueue::printHeadToTail(){
    Node *current=head;
    int c=0;
    while(current!=NULL){
        printf("%3d",current->elem);
        current=current->next;
        if((++c%20)==0)
            printf("\n");
    }
    if((c%20)!=0)
        printf("\n");
}
```

Current is a pointer, run for every element. But there is the key that ++c, plus the c first and return the value.

Main.cpp



```cpp
int main(){
    IQueue Q;
    int enqueue_count;
    int dequeue_count;
    int trial_count;

    int i, j;
    srand(time(NULL));
    trial_count = rand()%10+1;
    cout<<"Trial count:" <<trial_count;

    for(i=0; i<trial_count; i++){
        cout<<"\n\n>>> Trial " <<i+1<<": enqueue and dequeue operations\n";
        enqueue_count = rand()%100+1;
        cout<<"Enqueue "<<enqueue_count<<" elements to the queue.\n";
        for(j=0; j<enqueue_count; j++)
            Q.enqueue(rand()%100);
        cout<<"Current queue size: "<<Q.getSize();
        printf(" Content of queue from head to tail:\n");
        Q.printHeadToTail();
        dequeue_count = rand()%Q.getSize();
        cout<<"\n\nDequeue "<<dequeue_count<<" elements to the queue.\n";
        for(j=0; j<dequeue_count; j++)
            Q.dequeue();
        cout<<"Current queue size: "<<Q.getSize();
        cout<<". Content of queue from head to tail: \n";
        Q.printHeadToTail();
        cout<<"-----------------------------";
    }

    return 0;
}
```

Here is to print out the queue. First, produce the random number for trial count. And then run for out put the enqueue. At the same time, have to out put the random number for random times. Do it for enqueue and dequeue.