Student Name: 杨卓
Student ID: 2022141450295

高级语言程序设计-II
Assignment 6

1 Declaration

In file `linkedlist.h`, we declare these function which are in the documentation.

```cpp
#ifndef LINKED_LIST
#define LINKED_LIST

#include <iostream>
#include <initializer_list>

class LinkedList {
public:
    class Node {
    friend class LinkedList; // friend class
    public:
        Node();
        Node(double);
        Node *next;
        Node *previous;
        double getValue();
        void setValue(double);
        friend std::ostream& operator <<(std::ostream&, const Node&);
    private:
        double value;
    };
    LinkedList();
    LinkedList(const LinkedList&);
    LinkedList(std::initializer_list<double>);
    ~LinkedList();
    void push_back(double);
    void push_front(double);
    double pop_back();
    double pop_front();
    double back();
    double front();
    bool empty() const;
    void clear();
    void show();
    int getSize() const;
    void extend(const LinkedList&);
    double& operator[](const int&);
private:
    int N{0};
public:
    Node *head;
    Node *tail;
};

#endif
```

However, some functions or operators are not shown in the readme.md file, so we have to read the test program to find them. This causes compile errors when I think I have written all the functions.

For example, we need to overload the stream operator ('<<') for both `class LinkedList` and `class Node`, and use `std::initializer_list` to construct the class.

To overload the stream operator, we can use a friend class to access the private member variable `value` in `class Node` from `class LinkedList`.

Also, we should declare the functions `empty()` and `getSize()` as `const`. This way, const objects can use these functions. Only const functions can be called by const objects.

2 implementation

Implementation of these functions are in file `linkedlist.cpp`.

```
#include "../h/linkedlist.h"
#include <iomanip>
#include <iostream>
#include <initializer_list>

// Node:
LinkedList::Node::Node() {
    value = 0;
    next = previous = nullptr;
}

LinkedList::Node::Node(double v) {
    this->value = v;
    next = previous = nullptr;
}

double LinkedList::Node::getValue() {
    return this->value;
}

```

```cpp
void LinkedList::Node::setValue(double value) {
    this->value = value;
}

std::ostream &operator<<(std::ostream &out, const LinkedList::Node &node) {
    out << node.value;
    return out;
}

// LinkedList
// constructor
LinkedList::LinkedList() {
    N = 0;
    head = tail = new Node;
}

LinkedList::LinkedList(const LinkedList &list) {
    this->N = 0;
    this->head = this->tail = new Node;
    Node *p = list.head;
    while (p != list.tail) {
        this->push_back(p->getValue());
        p = p->next;
    }
}

LinkedList::LinkedList(std::initializer_list<double> list) {
    this->N = 0;
    this->head = this->tail = new Node;
    for (auto value : list) {
        this->push_back(value);
    }
}

// destructor
LinkedList::~LinkedList() {
    Node *p = head, *next;
    while (p != tail) {
        next = p->next;
        delete p;
        p = next;
    }
    delete tail;
}

// push
```

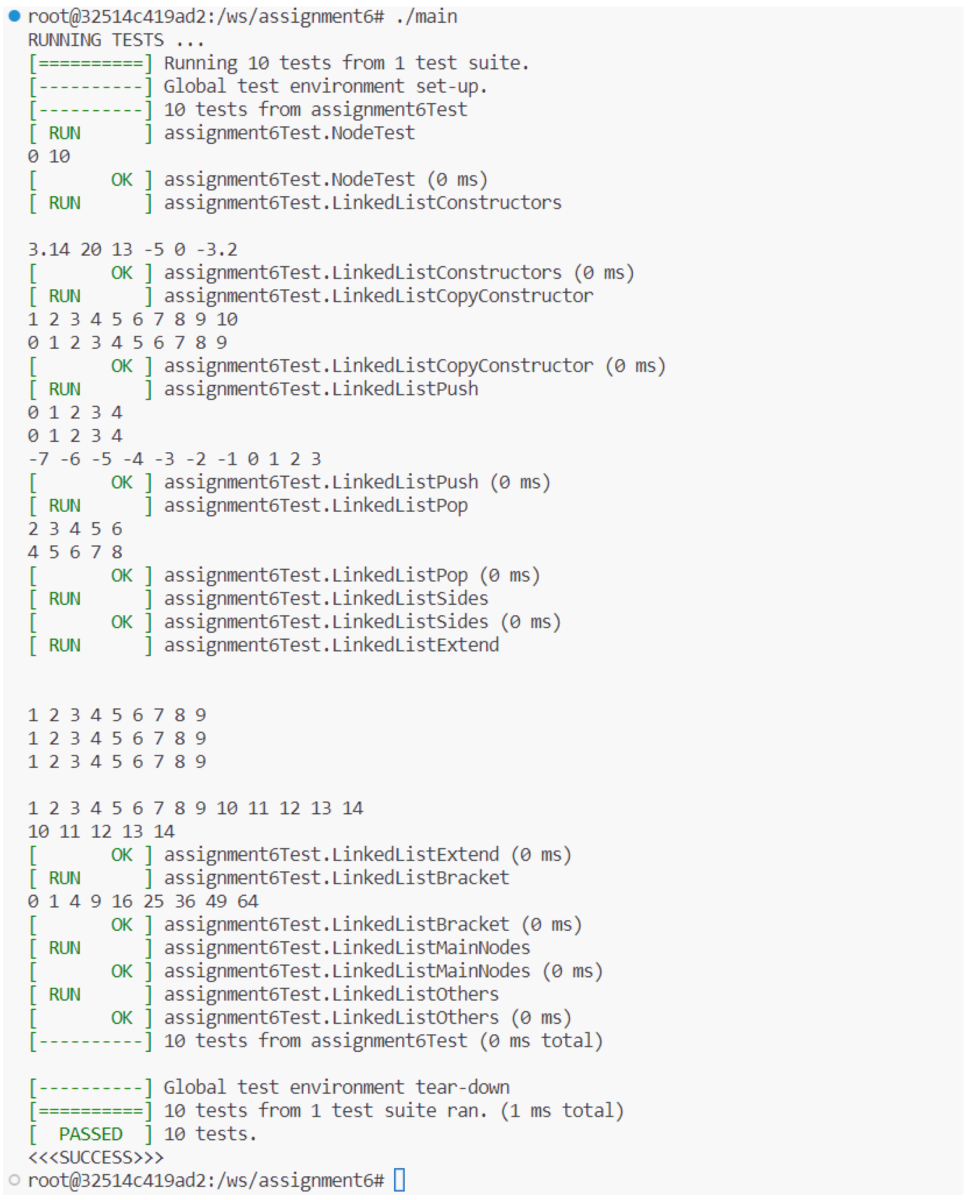
```cpp
void LinkedList::push_back(double value) {
    if (this->empty()) {
        head = new Node(value);
        head->next = tail;
        tail->previous = head;
    } else {
        Node *newnode = new Node(value);
        newnode->previous = tail->previous;
        newnode->next = tail;
        tail->previous->next = newnode;
        tail->previous = newnode;
        tail->value = value;
    }
    ++N;
}

void LinkedList::push_front(double value) {
    Node *newnode = new Node(value);
    newnode->next = head;
    head->previous = newnode;
    head = newnode;
    tail->setValue(tail->previous->getValue());
    ++N;
}

// pop
double LinkedList::pop_back() {
    if (this->empty())
        throw std::logic_error("Empty List!");
    Node *back =  tail->previous;
    Node *previous = back->previous;
    double back_value = back->getValue();
    previous->next = tail;
    tail->previous = previous;
    delete back;
    --N;
    if (!this->empty())
        tail->setValue(tail->previous->getValue());
    return back_value;
}

double LinkedList::pop_front() {
    if (this->empty())
        throw std::logic_error("Empty List!");
    double front_value = head->getValue();
    Node *next = head->next;
    delete head;
    head = next;
    --N;
    if (!this->empty())
        tail->setValue(tail->previous->getValue());
    return front_value;
}

// back&front value
double LinkedList::back() {
    if (this->empty())
        throw std::logic_error("Empty List!");
    return tail->previous->getValue();
}

double LinkedList::front() {
    if (this->empty())
        throw std::logic_error("Empty List!");
    return head->getValue();
}

// other information
bool LinkedList::empty() const {
    return head == tail;
}

void LinkedList::clear() {
    if (this->empty()) return;
    Node *p = head, *next;
    while (p != tail) {
        next = p->next;
        delete p;
        p = next;
    }
    head = tail;
    N = 0;
}

void LinkedList::show() {
    Node *p = head;
    while (p != tail) {
        std::cout << p->getValue() << " ";
        p = p->next;
    }
    std::cout << std::endl;
}
```

```cpp

int LinkedList::getSize() const {
    return this->N;
}

void LinkedList::extend(const LinkedList &list) {
    if (this->empty()) {
        this->N = 0;
        this->head = this->tail = new Node;
    }
    Node *p = list.head;
    while (p != list.tail) {
        this->push_back(p->getValue());
        p = p->next;
    }
}

double& LinkedList::operator[](const int &idx) {
    if (this->empty())
        throw std::logic_error("Empty List!");
    if (idx < 0 || idx >= N)
        throw std::logic_error("Out of range!");
    int index = 0;
    Node *p = head;
    while (p != tail) {
        if (idx == index) return p->value;
        ++index;
        p = p->next;
    }
    throw std::logic_error("Out of range!");
}
```

We should be careful when using pointers, as a small mistake can cause a run-time error.

We should throw an error when the parameters are invalid. This is not mentioned in the readme.md file.

3 Test result



```
● root@32514c419ad2:/ws/assignment6# ./main
RUNNING TESTS ...
[==========] Running 10 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 10 tests from assignment6Test
[ RUN      ] assignment6Test.NodeTest
0 10
[       OK ] assignment6Test.NodeTest (0 ms)
[ RUN      ] assignment6Test.LinkedListConstructors

3.14 20 13 -5 0 -3.2
[       OK ] assignment6Test.LinkedListConstructors (0 ms)
[ RUN      ] assignment6Test.LinkedListCopyConstructor
1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9
[       OK ] assignment6Test.LinkedListCopyConstructor (0 ms)
[ RUN      ] assignment6Test.LinkedListPush
0 1 2 3 4
0 1 2 3 4
-7 -6 -5 -4 -3 -2 -1 0 1 2 3
[       OK ] assignment6Test.LinkedListPush (0 ms)
[ RUN      ] assignment6Test.LinkedListPop
2 3 4 5 6
4 5 6 7 8
[       OK ] assignment6Test.LinkedListPop (0 ms)
[ RUN      ] assignment6Test.LinkedListSides
[       OK ] assignment6Test.LinkedListSides (0 ms)
[ RUN      ] assignment6Test.LinkedListExtend


1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9

1 2 3 4 5 6 7 8 9 10 11 12 13 14
10 11 12 13 14
[       OK ] assignment6Test.LinkedListExtend (0 ms)
[ RUN      ] assignment6Test.LinkedListBracket
0 1 4 9 16 25 36 49 64
[       OK ] assignment6Test.LinkedListBracket (0 ms)
[ RUN      ] assignment6Test.LinkedListMainNodes
[       OK ] assignment6Test.LinkedListMainNodes (0 ms)
[ RUN      ] assignment6Test.LinkedListOthers
[       OK ] assignment6Test.LinkedListOthers (0 ms)
[----------] 10 tests from assignment6Test (0 ms total)

[----------] Global test environment tear-down
[==========] 10 tests from 1 test suite ran. (1 ms total)
[  PASSED  ] 10 tests.
<<<SUCCESS>>>
○ root@32514c419ad2:/ws/assignment6#
```

图 1: test result