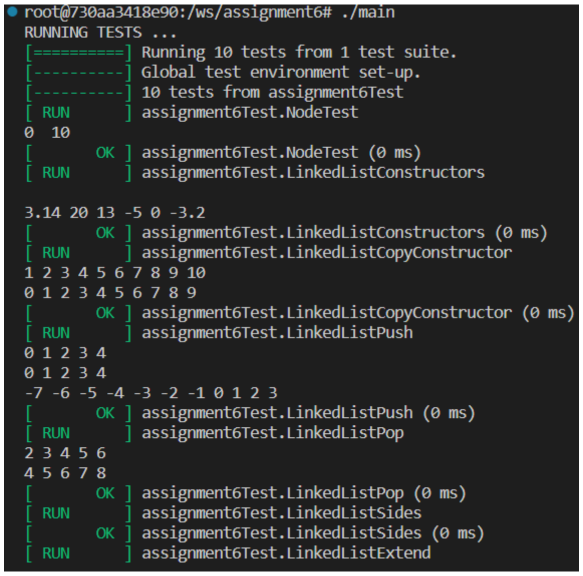
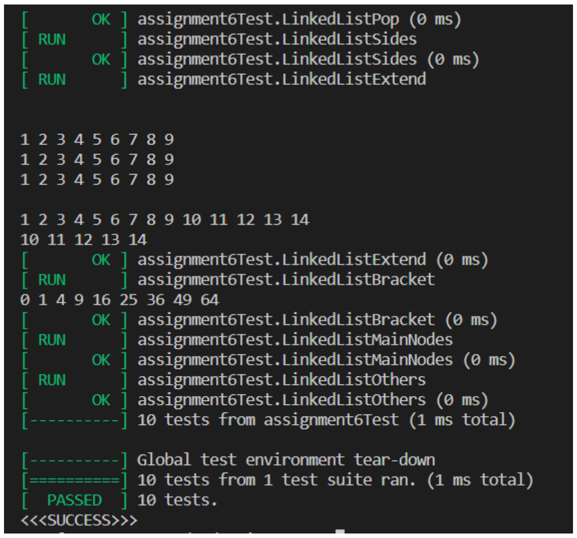
1.1 运行结果



```
root@730aa3418e90:/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
```



```
[       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 (1 ms total)

[----------] Global test environment tear-down
[==========] 10 tests from 1 test suite ran. (1 ms total)
[  PASSED  ] 10 tests.
<<<SUCCESS>>>
```

1.2 核心代码

初始化&析构

```
LinkedList:: Node:: Node():value(0),next(nullptr),previous(nullptr){}
LinkedList:: Node:: Node(double
_value):value(_value),next(nullptr),previous(nullptr){}
//初始化要注意初始化指针

LinkedList:: LinkedList():head(nullptr),tail(nullptr){}
//同样的，head和tail都要初始化（不然会给你随一个地址 就会有问题）
LinkedList:: LinkedList(std::initializer_list<double> a){
    head=tail=nullptr;
    for(auto x : a) push_back(x);
}
//用initializer_list初始化，和vector差不多，但是不能改变里面的值
LinkedList:: LinkedList(const LinkedList & Y){
    head=tail=nullptr;
    auto now=Y.head;
    while(now!=nullptr){
        push_back(now->getValue());
        now=now->next;
    }
}
//复制另一个链表，不能直接等于！
LinkedList:: ~LinkedList(){
    while(head!=nullptr){
        auto nxt=head->next;
        delete head;
        head=nxt;
    }
}
```

```
//析构函数需要把自己new出来的空间delete掉
```

链表的各种操作

```
void LinkedList:: push_back(double x){
    if(head==nullptr){
        head=tail=new Node(x);
        N++;
        return;
    }
    tail->next=new Node(x);//new一个节点出来，加入链表最后
    tail->next->previous=tail;
    tail=tail->next;
    N++;
}
void LinkedList:: push_front(double x){
    if(head==nullptr){
        head=tail=new Node(x);
        N++;
        return;
    }
    head->previous=new Node(x);//new一个节点出来，加入链表最前面
    head->previous->next=head;
    head=head->previous;
    N++;
}
void LinkedList:: pop_back(){
    if(tail==nullptr){ throw std::logic_error(""); return; }
    auto now=tail->previous;
    delete tail;//delete 最后的Node
    tail=now;
    if(tail!=nullptr) tail->next=nullptr;
    else head=tail=nullptr;
    N--;
}
void LinkedList:: pop_front(){
    if(head==nullptr){ throw std::logic_error(""); return; }
    auto now=head->next;//delete 最前面的Node
    delete head;
    head=now;
    if(head!=nullptr) head->previous=nullptr;
    else head=tail=nullptr;
    N--;
}
double LinkedList:: back(){
    if(tail==nullptr){ throw std::logic_error(""); return 0; }//记得throw error
    return tail->getValue();
}
double LinkedList:: front(){
    if(head==nullptr){ throw std::logic_error(""); return 0; }
    return head->getValue();
```

```
}
bool LinkedList:: empty(){//判断是否为空
    if(head==nullptr) return 1;
    return 0;
}
void LinkedList:: clear(){//清空 要清空间
    while(head!=nullptr){
        auto nxt=head->next;
        delete head;
        head=nxt;
    }
    head=tail=nullptr;
    N=0;
}
void LinkedList:: show(){//输出
    auto now=head;
    while(now!=nullptr){
        std::cout<<now->getValue()<<' ';
        now=now->next;
    }
    std::cout<<std::endl;
}
int LinkedList:: getSize(){
    return N;
}
void LinkedList:: extend(const LinkedList& Y){
    auto now=Y.head;
    if(tail==nullptr&&now!=nullptr){
        head=tail=new Node(now->getValue());
        now=now->next;
    }
    while(now!=nullptr){
        tail->next=new Node(now->getValue());
        tail->next->previous=tail;
        tail=tail->next;
        now=now->next;
    }
}
double& LinkedList:: operator[](int pos){
    auto now=head;
    if(pos>=0){//顺序
        while(pos--){
            if(now==nullptr){ throw std::logic_error(""); }
            now=now->next;
        }
    }
    else{//逆序
        now=tail;
        while((++pos)!=0){
            if(now==nullptr){ throw std::logic_error(""); }
            now=now->previous;
        }
    }
    if(now==nullptr) throw std::logic_error("");
    return (double&)(now->value);//返回一个引用
}
```