数据结构 2022秋 Lab1

> TA：张皓捷 19302010021 吴逸昕 19302010013

说明

本Lab主要关于第一周课上讲的链表等内容。

如无特殊情况，请使用C++完成本Lab。

建议使用CLion或Visual Studio Code或Visual Studio完成Lab。

任务

0.链表与数组的相互转换

在LinkedListVectorConverter.cpp中，实现toLinkedList和toVector方法。

toLinkedList方法应该读取传入的vector，并将vector中的数据按顺序转换成一个单链表，最后返回单链表的头节点。在创建单链表节点时，你可以使用new关键字。

toVector方法接收一个单链表的头节点，并将该单链表中的数据按顺序转换成一个vector，最后返回这个vector。如果头节点是NULL，toVector方法应该返回一个空的vector。

```
Node *LinkedListVectorConverter::toLinkedList(const std::vector<int> &v) {
    //TODO
    return nullptr;
}

std::vector<int> LinkedListVectorConverter::toVector(Node *head) {
    //TODO
    std::vector<int> v;
    return v;
}
```

1.链表重排列

在Problem1.cpp中，实现rearrangeNodes方法。

rearrangeNodes方法接收一个链表的头节点，并进行如下的处理。最后返回处理后的链表的头节点。

> 给定一个链表$a_1->a_2->a_3.....->a_n$，对链表进行重排列，使链表变为
> $a_1->a_n->a_2->a_{n-1}->a_3->a_{n-2}->.....$

例子1

输入：1->2->3->4->5->6，输出1->6->2->5->3->4

例子2

输入：1->2->3->4->5->6->7，输出1->7->2->6->3->5->4

注意：你的实现应该具有$O(1)$的空间复杂度$O(n)$的时间复杂度。n为原始链表长度。

```
Node *Problem1::rearrangeNodes(Node *head) {
    //TODO
    return head;
}
```

2.多项式相加

我们可以用单链表实现一个多项式：链表的每一个节点代表多项式中的一项；链表的节点中存储系数(coefficient)、指数(exponent)和下一项的指针(next)。**链表中的节点应该根据指数从大到小排列。**

```
class PolynomialTerm {
public:
    int coefficient{};
    int exponent{};
    PolynomialTerm *next{};

    PolynomialTerm(int coefficient, int exponent);
};
```

请你实现在Problem2.cpp中实现以下几个函数。

addTerm：在多项式中添加(系数,指数)=(coefficient,exponent)的项。如果指数为exponent的项已经存在，则需要合并同类项。

add：将两个多项式相加得到一个新的多项式，返回新的多项式的头节点。需要保障输入的两个多项式的链表结构保持不变。

toString：将多项式转换为形如 `3x^5-4x^4+8x^3+2x^2-9x+8+15x^-1` 的字符串表示。规则如下：

1. 不允许出现系数为0的项，例如 `3x^2` 是合法的输出，而 `3x^2+0x` 是不合法的输出
2. 当某项的系数为+1或-1时，应该省略系数，例如 `5x^2+x` 和 `9x^3-x` 是合法的输出，而 `6x^4+1x` 是不合法的输出
3. 当某项的次数为1或0时，应该省略次数，例如 `3x+5` 是合法的输出，而 `3x^1+5x^0` 是不合法的输出
4. 输出的多项式，应该按照指数从大到小排列；不允许出现具有相同指数的两项，例如 `8x^2` 是合法的输出，而 `3x^2+5x^2` 是不合法的输出
5. 如果多项式中没有任何一项，应该输出 `0`

free：释放链表中的所有节点

注意：在不考虑add和toString的返回值的占用空间的前提下，以上所有函数的实现均应具有$O(n)$的时间复杂度和$O(1)$的空间复杂度。

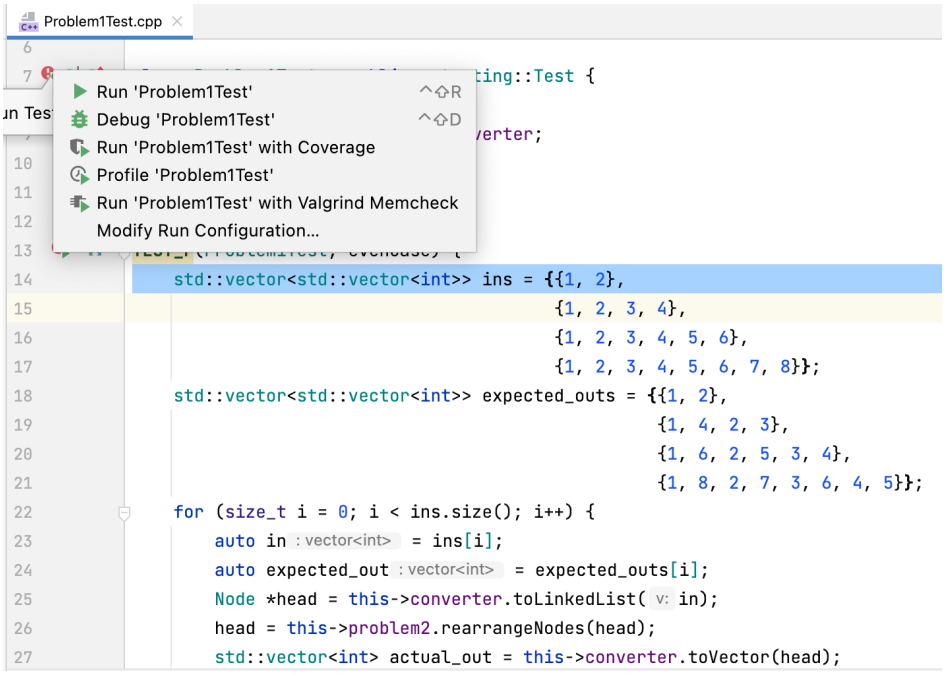
```
PolynomialTerm *Problem2::addTerm(PolynomialTerm *head, int coefficient, int exponent)
{
    //TODO
    return nullptr;
}

PolynomialTerm *Problem2::add(PolynomialTerm *head1, PolynomialTerm *head2) {
    //TODO
    return nullptr;
}

std::string Problem2::toString(PolynomialTerm *head) {
    //TODO
    return "";
}

void Problem2::free(PolynomialTerm *head) {
    //TODO
}
```

运行单元测试

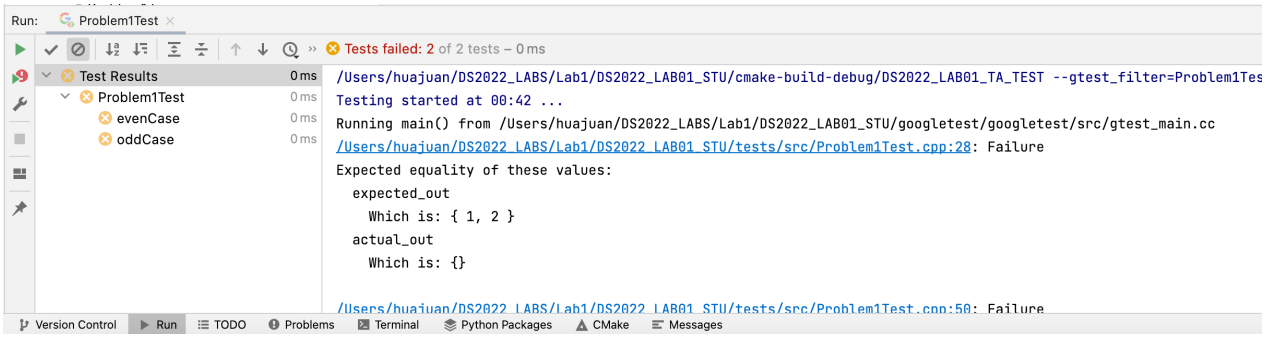
为了方便同学检验自己的程序是否正确，也方便助教批改，现编写了一些单元测试用例。

CLion

打开Problem1Test.cpp和Problem2Test.cpp，应该就能看到运行测试的按钮。

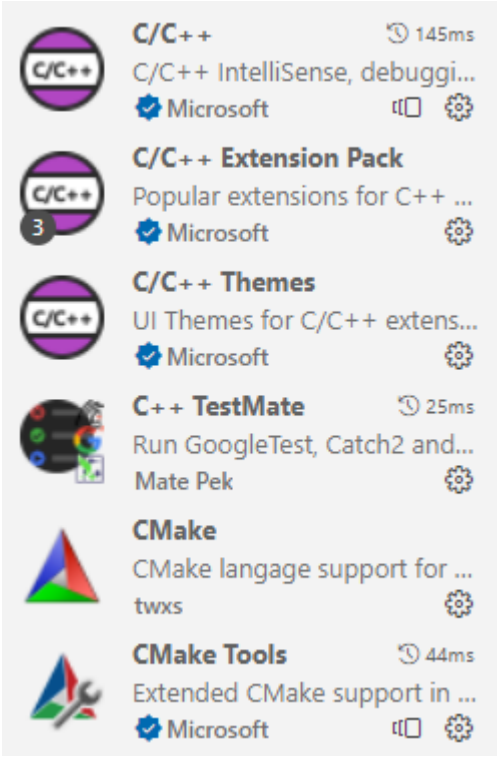


可以看到测试运行结果，实际输出值和期望输出值的差距。

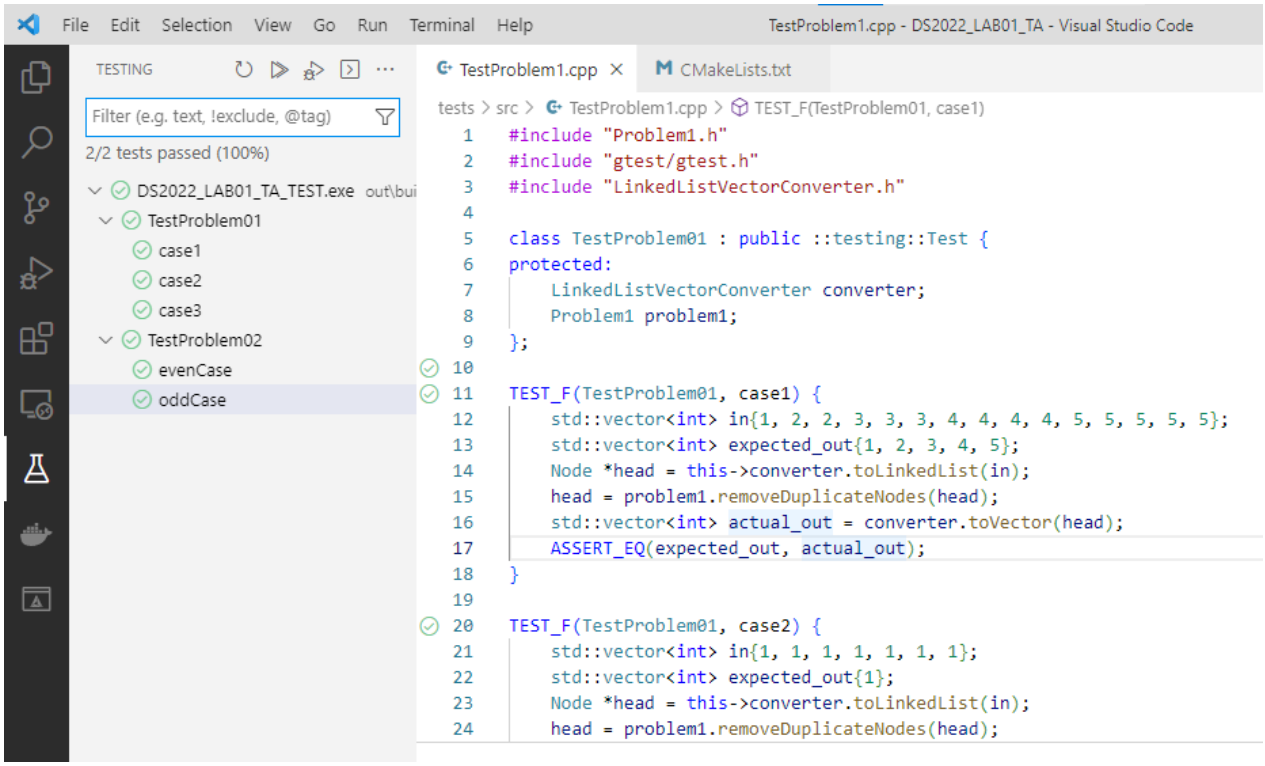


Visual Studio Code

需要在Visual Studio Code中安装以下几个扩展。



安装以后同样可以运行测试。



截止日期

2022年9月30日，星期五，23:59

提交

将你的整个项目打包成zip，上传到elearning。

文件命名为 学号-姓名-Lab1.zip，例如21302019999-花卷-Lab1.zip。