# 题目 A（继承）（60 分）

## 一、 题目描述

对生活中常见的规则物体进行描述刻画，其中包含描述所有规则物体的基类 Body，以及两种常见的规则物体的类：圆柱、圆锥以及三棱柱。具体要求如下：

1. 设计基类 Body，要求：
   a) 包含描述高度的属性：height
   b) 包含打印高度到控制台的函数 void printHeight();
2. 设计 Body 类的派生类圆柱类（Cylinder），要求：
   a) 包含描述底面半径的属性：radius
   b) 实现构造函数：Cylinder::Cylinder(double height, double radius)
3. 设计 Body 类的派生类圆锥类（Cone）,要求：
   a) 包含描述底面半径的属性：radius
   b) 实现构造函数：Cone::Cone(double height, double radius)
4. 设计 Body 类的派生类三棱柱类（ThreePrism）,要求：
   a) 包含描述底面三角形三条边长的属性：side_a, side_b, side_c
   b) 实现构造函数：ThreePrism::ThreePrism(double height, double side_a, double side_b, double side_c)
5. 所有派生类中包含打印底面面积到控制台的函数 void getArea().
6. 所有派生类中包含打印物体体积到控制台的函数 void getVolume().
7. 圆周率 pi 取 3.14.
8. 所有运算在运算过程中尽可能精确，在输出时则仅精确到 0.001.

## 二、 调用示例

示例代码：

```
Cylinder test1(4.0,5.0);

test1.getArea();

test1.getVolume();

test1.printHeight();

Cylinder test2(4.0,3.75);

test2.getArea();

test2.getVolume();

test2.printHeight();
```

示例运行结果：

```
78.500
314.000
4.000
44.156
176.625
4.000
```

## 三、 注意事项

注意输出与运算中的有效数字问题

## 四、 提交要求

1. 提交 8 个源码文件：Body.h、Body.cpp、Cylinder.h、Cylinder.cpp、ThreePrism.h、ThreePrism.cpp、Cone.h、Cone.cpp，直接打包成为 zip 格式的压缩包，不要添加任何其他目录。
2. 实现代码必须写在.cpp 文件中，声明必须写在.h 文件中。
3. 请严格按照给定的接口进行编码，否则无法调用测试样例。
4. 提交的源码文件中不要包含 main 函数，否则将无法通过编译。
5. 对要求实现的函数，除其指定功能外，不要尝试进行其他输入输出，否则无法通过测试。

# 题目 B（重载/模板）（20 分）

## 一、题目描述

STL 中的 vector 是一个封装了动态大小数组的顺序容器，它能够存放各种类型的对象。要求实现一个简易 Vector 类，不考虑动态扩容。

假设元素类型为 T，Vector 的接口要求如下：

```
typedef  T*  iterator;


iterator start; //动态数组入口，成员变量

size_t m_size; //当前数组大小，成员变量

size_t m_capcity; //Vector 所能容纳的最大元素数量，成员变量


Vector(size_t c); //初始化一个容量为 c 的 Vector，需要申请空间

~Vector(); //析构函数，清理空间

void push_back(T e); //在尾部增加一个元素

size_t size();

iterator begin(); //返回第一个元素的迭代器

iterator end(); //返回最后一个元素后面一个位置的迭代器，即[begin, end)


//在 it 前插入一个元素，返回的迭代器指向插入元素

iterator insert(iterator it, T e);


//删除 it 指向的元素，返回的迭代器指向删除元素的下一个元素

iterator erase(iterator it);


//TODO：重载 [] 操作，返回相应元素的引用
```

## 二、注意事项：

要求使用类模板，**支持所有的基本的数据类型**，请注意 Vector 类的声明，不要使用 STL。

## 三、提交要求

1 提交一个源码文件，Vector.h，直接打包成 zip 压缩格式的压缩包。不要添加任何其他目录。

2 实现和声明都写在 h 头文件中，文件编码格式为 utf-8。

3 请严格按照给定的接口进行编码，否则无法调用测试用例。

4 提交的源码文件中，不能包含 main 函数，否则无法编译通过。

## 四、具体实例

调用示例：

```
Vector<int> v(10); //最多可容纳 10 个元素
v.push_back(1);
v.push_back(2);
v.push_back(3);
Vector<int>::iterator it = v.begin();
for(it = v.begin(); it != v.end(); it++) {
    cout << *it << endl;
}
v.insert(v.begin(), 0);
v.erase(v.end() - 1);
for(size_t i = 0; i < v.size(); i++) {
    cout << v[i] << endl;
}
```

结果：

1

2

3

0

1

2

# 题目 C（链表）（20 分）

## 一、题目描述

1. 已知 f 为单链表的表头指针，链表中存储的都是整型数据，试实现下列算法：

   (1) 求链表中的最大整数

   (2) 求链表的结点个数

   (3) 求所有整数的平均值

2. 接口要求

```
#include <iostream>    //定义在头文件"RecurveList.h"中
using namespace std;
class List;
class ListNode {                    //链表结点类
    friend class List;
private:
    int data;           //结点数据
    ListNode *link;     //结点指针
    ListNode(const int item) : data(item), link(NULL) { } //构造函数
};
class List {            //链表类
private:
    ListNode *first, *current;
    int tmp;//临时变量
    int Max(ListNode *f);
    int Num(ListNode *f);
    float Avg(ListNode *f, int& n);
public:
    List() : first(NULL), current(NULL),tmp(0) { }        //构造函数
    ~List() { }                 //析构函数
    ListNode* NewNode(const int item);//创建链表结点，其值为 item
    void NewList(const int retvalue); //建立链表，以输入 retvalue 结束
    void PrintList();           //输出链表所有结点数据
    int GetMax() { return Max(first); }       //求链表所有数据的最大值
    int GetNum() { return Num(first); }       //求链表中数据个数
    float GetAvg() { return Avg(first,tmp); }     //求链表所有数据的平均值
};
```

## 二、注意事项

1）构造函数和析构函数已实现

2）不考虑链表为空的情况，即输入第一个元素 value != retvalue

3）输出链表函数 void PrintList();为元素间以一个空格隔开，不要有换行。

## 三、提交要求

提交两个源码文件，RecurveList.h 和 RecurveList.cpp 直接打包成 zip 格式的压缩包，不要添加任何其他目录。

实现代码必须写在 cpp 文件中，声明必须写在.h 头文件中。

请严格按照给定的接口进行编码，否则无法调用测试用例。

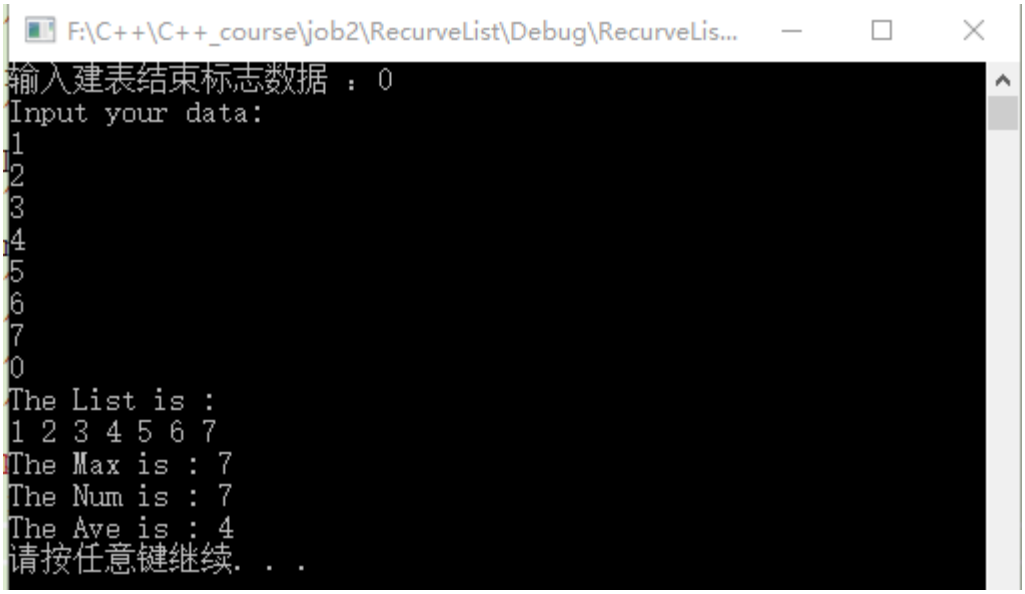
提交的源码文件中不需要包含 main 函数，否则将无法通过编译。

## 四、具体实例

调用代码：

```
int main()
{
    List test;
    int finished;
    cout << "输入建表结束标志数据 : ";
    cin >> finished;           //输入建表结束标志数据
    cout << "Input your data: " << endl;       //提示
    test.NewList(finished);        //建立链表
    cout << "The List is : " << endl;
    test.PrintList();          //打印链表
    cout << endl;
    cout << "The Max is : " << test.GetMax() << endl;
    cout << "The Num is : " << test.GetNum() << endl;
    cout << "The Ave is : " << test.GetAvg() << endl;
    return 0;
}
```

代码输出：


F:\C++\C++_course\job2\RecurveList\Debug\RecurveLis...
输入建表结束标志数据 ： 0
Input your data:
1
2
3
4
5
6
7
0
The List is :
1 2 3 4 5 6 7
The Max is : 7
The Num is : 7
The Ave is : 4
请按任意键继续. . .