实践1: 利用大模型将python程序转换成C++程序

python程序在群文件: "f001_Python代码用例集.zip" 中, 编号为1-163的163个程序需要认领, 请同学们按照次序每人认领5个python程序进行实验,

认领表格链接为: https://kdocs.cn/l/cbHBEU8wx9zf

请各位同学将第三列继续向后补完(满163之后循环从0开始, 例如 161-163-0-1, 2-6, ...), 并将自己的学号和姓名填写在前面两列中的某一行.

请大家将认领到的程序, 尝试借助大语言模型将本文件夹中的 Python 代码转换为 C++ 的代码。应该提交: 1.原始python代码, 2.大模型输出的C++代码 (标注由哪个模型输出, 以及驱动大模型的prompt), 3.以及手动修正后的C++代码(需要**通过编译, 且可执行, 执行结果正确**.)

输入的 Python 代码均由两部分组成，主要功能函数（函数名不确定，函数中含有功能说明注释）与测试函数（`check(candidate)` 函数）; 手动修正后的 C++ 代码由三部分组成，包括主要功能函数(函数名不确定), 功能说明注释(**请手动将功能说明注释同样补充在功能函数中, 粘贴过去即可**), 与主函数(main`函数，用于调用主要功能函数来进行测试，不需要包含原有`check(candidate)`中的所有用例)。

可以自行设计与大语言模型交互的 `prompt`，这里给出一些建议, 例如:

````
请将下列python代码改成语义上等价的C++代码 (Please transfer the following python code
into semantically equivalent C++ code):
```python
这里粘贴python原始的代码
```
````

或者针对注释中的说明来构建prompt, 以 `000_has_close_elements.py` 为例:

```
请按照以下要求来构造C++代码 (Please write the C++ code that following the
descriptions below):
""" For a given list of integers, return a tuple consisting of a sum and a
product of all the integers in a list.
    Empty sum should be equal to 0 and empty product should be equal to 1.
    >>> sum_product([])
    (0, 1)
    >>> sum_product([1, 2, 3, 4])
    (10, 24)
"""
```

以上prompt仅供参考, 同学们如果有好的prompt可以尽情发挥, 如果prompt不好, 那么后面修正C++代码的工作量就会比较大, 最坏情况下这个C++的转换就全部是手工完成的. 即使如此, 只要最终手动修正的C++代码完全正确, 这部分实验就会给分.

最终得分情况**基本仅考虑最终手动修正的C++代码**, 大模型仅仅是提供一个工具.

以下为转换样例：

Python 原始代码用例 `000_has_close_elements.py`：

```python
from typing import List, Tuple

def sum_product(numbers: List[int]) -> Tuple[int, int]:
    """ For a given list of integers, return a tuple consisting of a sum and a
product of all the integers in a list.
    Empty sum should be equal to 0 and empty product should be equal to 1.
    >>> sum_product([])
    (0, 1)
    >>> sum_product([1, 2, 3, 4])
    (10, 24)
    """
    assert all([isinstance(v, int) for v in numbers]), "invalid inputs" #
$_CONTRACT_$
    s, p = 0, 1
    for number in numbers:
        s += number
        p *= number
    return s, p

def check(candidate):
    assert candidate([]) == (0, 1)
    assert candidate([1, 1, 1]) == (3, 1)
    assert candidate([100, 0]) == (100, 0)
    assert candidate([3, 5, 7]) == (3 + 5 + 7, 3 * 5 * 7)
    assert candidate([10]) == (10, 10)

check(sum_product)
```

最终手动修正的代码用例 `000_has_close_elements.cpp`，主函数中只需包含一个测试用例即可：

```cpp
#include <iostream>
#include <vector>
#include <tuple>
#include <cassert>

std::tuple<int, int> sum_product(const std::vector<int>& numbers) {
    /*
    For a given list of integers, return a tuple consisting of a sum and a
product of all the integers in a list.
    Empty sum should be equal to 0 and empty product should be equal to 1.
    */
    if (numbers.empty()) {
        return std::make_tuple(0, 1);
    }

    int sum = 0, product = 1;

    for (int number : numbers) {
        sum += number;
        product *= number;
    }

    return std::make_tuple(sum, product);
```

```cpp
}

int main() {
    std::tuple<int, int> result = sum_product({});
    assert(std::get<0>(result) == 0);
    assert(std::get<1>(result) == 1);
    return 0;
}
```