中山大学本科生期中考试

考试科目：《程序设计II实验》

学年学期：2024-2025 学年第 2 学期

开课单位：计算机学院

考试方式：上机闭卷

考试时间：100 分钟

【A4】复制构造员工类

题目描述

以下有一个 `Employee` 类，这个类的对象会记录员工的工号 `id` 和姓名 `name` 信息，需要完成构造函数，拷贝构造函数和析构函数来完成深拷贝。 员工数量由输入的第一行给出，具体的 `id` 和 `name` 信息由后续的输入给出。

Employee 类

```
class Employee {
public:
    Employee(int* id, string* name, int len);
    Employee(const Employee& other);
    ~Employee();

    void employeePrint() {
        for(int i = 0; i < size; i++) {
            cout << "|" << id_list[i] << "|" << name_list[i] << "|" << endl;
        }
    }
private:
    int* id_list;
    string* name_list;
    int size;
};
```

样例输入

```
3
1001
1002
1003
Tony
Kimmy
Ming
```

样例输出

```
Employee1 is:
|1001|Tony|
|1002|Kimmy|
|1003|Ming|
Employee2 is:
|1001|Tony|
|1002|Kimmy|
|1003|Ming|
```

`main.cpp`：

```
#include "Employee.h"

int main()
{
    int n;
    cin >> n;
    int* id = new int[n];
    string* name = new string[n];

    for(int i = 0; i < n; i++)
    {
        cin >> id[i];
    }
    for(int i = 0; i < n; i++)
    {
        cin >> name[i];
    }
    Employee emp1(id,name,n);
    Employee emp2(emp1);

    cout << "Employee1 is: " << endl;
    emp1.employeePrint();
    cout << "Employee2 is: " << endl;
    emp2.employeePrint();
    return 0;
}
```

`Employee.h`：

```
#ifndef EMPLOYEE_HPP
#define EMPLOYEE_HPP
#include <iostream>
#include <string>
using namespace std;


class Employee{
public:
    Employee(int* id, string* name, int len);
    Employee(const Employee& other);
    ~Employee();

    void employeePrint()
    {
        for(int i = 0; i < size; i++)
        {
            cout << "|" << id_list[i] << "|" <<name_list[i] << "|" << endl;
        }
    }
private:
    int* id_list;
    string* name_list;
    int size;
};

#endif
```

`Employee.cpp`：（答案，仅供参考）

编译检查 20%，标准测试 20%，随机测试 60%

```
#include "Employee.h"

Employee::Employee(int* id, string* name, int len):size(len) {
    id_list = new int[size];
    name_list = new string[size];

    for (int i=0; i<size; i++) {
        id_list[i] = id[i];
        name_list[i] = name[i];
    }
}

Employee::Employee(const Employee& other) {
    size = other.size;
    id_list = new int[size];
    name_list = new string[size];

    for (int i=0; i<size; i++) {
        id_list[i] = other.id_list[i];
        name_list[i] = other.name_list[i];
    }
}

Employee::~Employee() {
    delete[] id_list;
    delete[] name_list;
}
```