Turnin 作業 11

- Turnin Code: cpp.hw11
- Due Date: 6/8 Monday 23:59:00 (midnight) **Hard Deadline**

本次作業為本學期最後一次作業！成功是屬於堅持到底的人們！加油！

繳交方式說明

本次作業繳交將以資料夾的形式繳交，需要為每一題建立一個資料夾（資料夾名稱為該題題目前方之代號，第一題為p1，第二題為p2，餘以此類推）。

繳交說明可參考【Turnin 作業 4】中 p6 到 p11

任何未依照正確繳交格式的檔案將以 0 分計。

本文使用「 」及「 」代表「空白字元」與Enter 換行字元」，並且將使用者輸入的部份使用灰階方式顯示。另外，題目的執行結果中，如果出現「(」、「)」、「:」、「;」、「.」與「,」等符號，皆為英文半形！

---

p1 學生類別（動態多型與覆寫）

接續 cpp.hw10 p1本題將要求同學們用動態多型的方式輸出相同內容。本題改用虛擬函式 virtual 關鍵字），讓在不同物件執行相同名稱相同引數的函式時，不再需要透過強制轉型的方式就可直接呼叫自己的版本；也就是動態多型Dynamic Polymorphism的效果。

Student 類別相關定義請參考下列的 student.h 程式（請依據檔名旁的路徑至 ws 取得程式碼）：

```
#include <iostream>
#include <string>
using namespace std;

enum StudentType
{
    Normal,
    Foreign,
```

```
    Local,
    LocalParttime
};

class Student
{
public:
    string name;
    StudentType type;
    Student();
    virtual void showInfo() = 0;
};
```

ForeignStudent 類別相關定義請參考下列的 ForeignStudent.h 程式（請依據檔名旁的路徑至 ws 取得程式碼）：

```
#include "Student.h"

class ForeignStudent : public Student
{
public:
    string nationality;
    ForeignStudent();
    void showInfo() override;
};
```

LocalStudent 類別相關定義請參考下列的 LocalStudent.h 程式（請依據檔名旁的路徑至 ws 取得程式碼）：

```
#include "ForeignStudent.h"

class LocalStudent : public Student
{
public:
    LocalStudent();
    void showInfo() override;
};
```

LocalParttimeStudent 類別相關定義請參考下列的 LocalParttimeStudent.h 程式（請依據檔名旁的路徑至 ws 取得程式碼）：

```
#include "LocalStudent.h"

class LocalParttimeStudent : public LocalStudent
{
public:
    LocalParttimeStudent();
    void showInfo() override;
```

```
};
```

請參考下列的 main.cpp 程式（請依據檔名旁的路徑至 ws 取得程式碼）：

```
#include "LocalParttimeStudent.h"
using namespace std;
int main()
{
    int amt = 0;
    cout << "Input student quantity: ";
    cin >> amt;
    Student *csie2c[amt];
    int i;
    for (int i = 0; i < amt; i++)
    {
        char student_type;
        cout << "Input student type(F/L/P): ";
        cin >> student_type;

        switch (student_type)
        {
            case 'F':
                csie2c[i] = new ForeignStudent;
                cout << "[ForeignStudent] created." << endl;
                break;
            case 'L':
                csie2c[i] = new LocalStudent;
                cout << "[LocalStudent] created." << endl;
                break;
            case 'P':
                csie2c[i] = new LocalParttimeStudent;
                cout << "[LocalParttimeStudent] created." << endl;
                break;
        }
    }
    cout << "----------------------" << endl;

    for (int i = 0; i < amt; i++)
    {
        csie2c[i]->showInfo();
    }
    return 0;
}
```

檔案名稱	**檔案用途與撰寫要求**	**-**
main.cpp	主程式	題目提供原始碼
Student.h	Student 類別的定義	題目提供原始碼

檔案名稱	**檔案用途與撰寫要求**	**-**
Student.cpp	Student 類別的實作	需繳交
ForeignStudent.h	ForeignStudent 類別的定義	題目提供原始碼
ForeignStudent.cpp	ForeignStudent 類別的實作	需繳交
LocalStudent.h	LocalStudent 類別的定義	題目提供原始碼
LocalStudent.cpp	LocalStudent 類別的實作	需繳交
LocalParttimeStudent.h	LocalParttimeStudent 類別的定義	題目提供原始碼
LocalParttimeStudent.cpp	LocalParttimeStudent 類別的實作	需繳交

請完成名為 **Student.cpp**、**ForeignStudent.cpp**、**LocalStudent.cpp** 與 **LocalParttimeStudent.cpp** 的 C++ 程式。

本題的相關程式將使用以下的 Makefile 進行編譯：

```
all: main.cpp LocalParttimeStudent.o LocalStudent.o ForeignStudent.o
Student.o
    c++ main.cpp LocalParttimeStudent.o LocalStudent.o ForeignStudent.o
Student.o
LocalParttimeStudent.o: LocalParttimeStudent.cpp LocalParttimeStudent.h
LocalStudent.h ForeignStudent.h Student.h
    c++ -c LocalParttimeStudent.cpp
LocalStudent.o: LocalStudent.cpp LocalStudent.h ForeignStudent.h Student.h
    c++ -c LocalStudent.cpp
ForeignStudent.o: ForeignStudent.cpp ForeignStudent.h Student.h
    c++ -c ForeignStudent.cpp
Student.o: Student.cpp Student.h
    c++ -c Student.cpp
clean:
    rm -f *.o *~ *.*~ a.out*
```

此題的執行結果可參考如下：

```
[3:23▲user@ws▲p1]./a.out↵
Input▲student▲quantity:▲5↵
Input▲student▲type(F/L/P):▲F↵
[ForeignStudent]▲created.↵
Input▲student▲type(F/L/P):▲L↵
[LocalStudent]▲created.↵
Input▲student▲type(F/L/P):▲P↵
[LocalParttimeStudent]▲created.↵
Input▲student▲type(F/L/P):▲F↵
[ForeignStudent]▲created.↵
Input▲student▲type(F/L/P):▲P↵
[LocalParttimeStudent]▲created.↵
----------↵
I'm▲a▲foreign▲student.↵
I'm▲a▲local▲student.↵
```

```
I'm▲a▲local▲part-time▲student.↵
I'm▲a▲foreign▲student.↵
I'm▲a▲local▲part-time▲student.↵
[3:23▲user@ws▲p1]
```

- 本題相關的程式碼路徑已註明於檔名右側，同學們可以透過路徑複製到自己的家目錄。
- 類別撰寫要求請參考上方表格
- 本題若有使用浮點數的需求，請使用 double 型態。
- 本題應繳交檔案如下：
  - Student.cpp
  - ForeignStudent.cpp
  - LocalStudent.cpp
  - LocalParttimeStudent.cpp

p2 銀行帳戶類別實作（動態多型與覆寫）

接 cpp.hw10 p3請用動態多型的方式作答。

BankAccount 類別相關定義請參考下列的 bank.h 程式（請依據檔名旁的路徑至 ws 取得程式碼）：

```
#include <iostream>
#include <string>
using namespace std;

class BankAccount
{
protected:
    string name;
    int balance;
    struct
    {
        string record;
        int count = 1;
    } transaction;

public:
    BankAccount(string name_str);
    int get_balance();
    void set_balance(int new_balance);
    string get_name();
    void write_record(string record);
    void show_record();
    void save(int amount);
```

```
    virtual void show_balance();
    virtual void withdraw(int amount);
    virtual void transfer(int amount, BankAccount &recipient);
};
```

SavingAccount 類別相關定義請參考下列的 saving_account.h 程式（請依據檔名旁的路徑至 ws 取得程式碼）：

```
#include "bank.h"

class SavingAccount : public BankAccount
{
private:
    double deposit_interest_rate = 0.016;
    int handling_fee = 10;

public:
    SavingAccount(string name_str);
    void calculate_interest(int year);
    void show_balance() override;
    void withdraw(int amount) override;
    void transfer(int amount, BankAccount &recipient) override;
};
```

請參考下列的 main.cpp 程式（請依據檔名旁的路徑至 ws 取得程式碼）：

```
#include "saving_account.h"
using namespace std;

int main()
{
    BankAccount *customer[3];
    BankAccount Justin("Justin");
    BankAccount Amy("Amy");
    SavingAccount Phineas("Phineas");
    customer[0] = &Justin;
    customer[1] = &Amy;
    customer[2] = &Phineas;

    cout << "[1] ------------------------" << endl;
    customer[0]->show_balance();
    customer[1]->show_balance();

    cout << "[2] ------------------------" << endl;
    customer[1]->save(4090);
    customer[1]->save(0);
    customer[1]->save(-216);
```

```
    cout << "[3] ------------------------" << endl;
    customer[1]->transfer(1080, Justin);
    customer[1]->transfer(4050, Justin);
    customer[1]->transfer(0, Justin);
    customer[1]->transfer(-512, Justin);

    cout << "[4] ------------------------" << endl;
    customer[1]->withdraw(12000);
    customer[1]->withdraw(3000);
    customer[1]->withdraw(0);

    cout << "[5] ------------------------" << endl;
    customer[0]->show_balance();
    customer[1]->show_balance();

    cout << "[6] ------------------------" << endl;
    customer[1]->show_record();

    cout << "[7] ------------------------" << endl;
    customer[2]->show_balance();
    customer[2]->save(10000);
    customer[2]->show_balance();
    ((SavingAccount *)customer[2])->calculate_interest(10); // for 10 years
    customer[2]->show_balance();

    cout << "[8] ------------------------" << endl;
    customer[2]->withdraw(450);
    customer[2]->show_balance();

    cout << "[9] ------------------------" << endl;
    customer[2]->show_balance();
    customer[2]->transfer(10, Amy);
    customer[2]->show_balance();
    customer[1]->show_balance();

    cout << "[10] ------------------------" << endl;
    customer[2]->withdraw(11231);
    customer[2]->show_balance();
    customer[2]->transfer(11231, Justin);
    customer[2]->show_balance();
    customer[2]->withdraw(11230);
    customer[2]->show_balance();

    cout << "[11] ------------------------" << endl;
    customer[2]->show_record();

    return 0;
}
```

鳳梨銀行 SavingAccount 固有規則：

1. 本利和的計算以**年**為單位且年利率為 1.6%：$期末本利和 = 期初本金金額 \times (1 + 每期利率)^{期數}$，計算結果請無條件捨棄小數。

邪惡的鳳梨銀行希望 SavingAccount 使用者將錢持續存在銀行裡面而使用了複利計算本利合，但是銀行發現複利的計算方式沒辦法吸引對於那些存款較少的使用者持續讓錢儲存在銀行，因此頒訂了新規定：

1. 不論提款多少錢，必須支付 10 元的手續費費用。提款時，$(欲提領金額 + 手續費的金額) \lt 帳戶內的餘額$ 時，相當於沒有足夠的錢進行交易導致交易失敗。
2. 轉帳同提款的新規則。

上述規定僅限制於 SavingAccount 使用者。

請用動態多型的方式完成本題。

檔案名稱	**檔案用途與撰寫要求**	**-**
main.cpp	主程式	題目提供原始碼
bank.h	BankAccount 類別的定義	題目提供原始碼
bank.cpp	BankAccount 類別的實作	需繳交
saving_account.h	SavingAccount 類別的定義	題目提供原始碼
saving_account.cpp	SavingAccount 類別的實作	需繳交

請完成名為 **bank.cpp** 與 **saving_account.cpp** 的 C++ 語言程式，相關檔案撰寫要求已詳述於上方表格。

註：本題輸出請勿沿用上次作業的 ANSI 色碼（用於控制終端機文字樣式），否則視為錯誤

本題將使用以下的 Makefile 進行編譯：

```
all: main.cpp bank.o saving_account.o bank.h saving_account.h
    c++ main.cpp bank.o saving_account.o
saving_account.o: bank.h saving_account.cpp saving_account.h
    c++ saving_account.cpp -c
bank.o: bank.cpp bank.h
    c++ bank.cpp -c
clean:
    rm -f *.o *~ *.*~ a.out*
```

此題的執行結果可參考如下：

```
[3:23▲user@ws▲p2]./a.out↵
[1]▲----------------------↵
Justin's▲current▲balance:▲0↵
Amy's▲current▲balance:▲0↵
[2]▲----------------------↵
[▲SUCCESS▲]▲Amy▲saved▲\$4090↵
[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲save▲\$0↵
[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲save▲\$-216↵
[3]▲----------------------↵
```

[▲SUCCESS▲]▲Amy▲transfered▲\$1080▲to▲Justin↵
[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲transfer▲\$4050▲to▲Justin↵
[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲transfer▲\$0▲to▲Justin↵
[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲transfer▲\$-512▲to▲Justin↵
[4]▲------------------------↵
[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲withdraw▲\$12000↵
[▲SUCCESS▲]▲Amy▲withdrew▲\$3000↵
[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲withdraw▲\$0↵
[5]▲------------------------↵
Justin's▲current▲balance:▲1080↵
Amy's▲current▲balance:▲10↵
[6]▲------------------------↵
Amy's▲transaction▲record:▲↵
1▲[▲SUCCESS▲]▲Amy▲saved▲\$4090↵
2▲[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲save▲\$0↵
3▲[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲save▲\$-216↵
4▲[▲SUCCESS▲]▲Amy▲transfered▲\$1080▲to▲Justin↵
5▲[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲transfer▲\$4050▲to▲Justin↵
6▲[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲transfer▲\$0▲to▲Justin↵
7▲[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲transfer▲\$-512▲to▲Justin↵
8▲[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲withdraw▲\$12000↵
9▲[▲SUCCESS▲]▲Amy▲withdrew▲\$3000↵
10▲[▲▲▲FAIL▲▲]▲Amy▲failed▲to▲withdraw▲\$0↵
[7]▲------------------------↵
Phineas's▲current▲balance:▲0▲(Savings▲Account)↵
[▲SUCCESS▲]▲Phineas▲saved▲\$10000↵
Phineas's▲current▲balance:▲10000▲(Savings▲Account)↵
Phineas▲accrued▲interest▲for▲10▲yr(s).↵
Phineas's▲current▲balance:▲11720▲(Savings▲Account)↵
[8]▲------------------------↵
[▲SUCCESS▲]▲Phineas▲withdrew▲\$450▲with▲\$10▲handling▲fee↵
Phineas's▲current▲balance:▲11260▲(Savings▲Account)↵
[9]▲------------------------↵
Phineas's▲current▲balance:▲11260▲(Savings▲Account)↵
[▲SUCCESS▲]▲Phineas▲transfered▲\$10▲to▲Amy▲with▲\$10▲handling▲fee↵
Phineas's▲current▲balance:▲11240▲(Savings▲Account)↵
Amy's▲current▲balance:▲20↵
[10]▲------------------------↵
[▲▲▲FAIL▲▲]▲Phineas▲failed▲to▲withdraw▲\$11231↵
Phineas's▲current▲balance:▲11240▲(Savings▲Account)↵
[▲▲▲FAIL▲▲]▲Phineas▲failed▲to▲transfer▲\$11231▲to▲Justin↵
Phineas's▲current▲balance:▲11240▲(Savings▲Account)↵
[▲SUCCESS▲]▲Phineas▲withdrew▲\$11230▲with▲\$10▲handling▲fee↵
Phineas's▲current▲balance:▲0▲(Savings▲Account)↵
[11]▲------------------------↵
Phineas's▲transaction▲record:▲↵
1▲[▲SUCCESS▲]▲Phineas▲saved▲\$10000↵
2▲Phineas▲accrued▲interest▲for▲10▲yr(s).↵
3▲[▲SUCCESS▲]▲Phineas▲withdrew▲\$450▲with▲\$10▲handling▲fee↵

```
4▲[▲SUCCESS▲]▲Phineas▲transfered▲\$10▲to▲Amy▲with▲\$10▲handling▲fee↵
5▲[▲▲▲FAIL▲▲]▲Phineas▲failed▲to▲withdraw▲\$11231↵
6▲[▲▲▲FAIL▲▲]▲Phineas▲failed▲to▲transfer▲\$11231▲to▲Justin↵
7▲[▲SUCCESS▲]▲Phineas▲withdrew▲\$11230▲with▲\$10▲handling▲fee↵
[3:23▲user@ws▲p2]
```

- 本題相關的程式碼路徑已註明於檔名右側，同學們可以透過路徑複製到自己的家目錄。
- 本題輸出請勿沿用上次作業的 ANSI 色碼（用於控制終端機文字樣式），否則視為錯誤。
- 類別撰寫要求請參考上方表格。
- 本題應繳交檔案如下：
  - bank.cpp
  - saving_account.cpp

p3 大海撈針

「/」 運算子在大多數的程式語言代表除法，而除法是數學四則運算之一，它的意義是將一個數平均分成若干分的方法。本題為 「/」 運算子加入一個新定義：在一個字串（str）裡**子字串（needle）出現的次數**。該運算子使用方式為

```
string str = "ABABABABC";
string needle = "AB";
int number = str/needle;
cout << number << endl; // output should be 4
```

請參考下列的 main.cpp 程式（請依據檔名旁的路徑至 ws 取得程式碼）：

```
#include "operator.h"
#include <iostream>
using namespace std;

int main()
{
    string str;
    string needle;
    getline(cin, str);
    getline(cin, needle);

    cout << "Paragraph: " << endl << str << endl << endl;
    cout << "Needle: [" << needle << "]" << endl << endl;
    cout << "Occurrences of needle: " << str / needle << endl;
}
```

檔案名稱	檔案用途與撰寫要求	-
main.cpp	主程式	題目提供原始碼
operator.h	「/」 運算子多載的定義	需繳交
operator.cpp	「/」 運算子多載的實作	需繳交

請完成名為 **operator.h** 與 **operator.cpp** 的 C++ 語言程式，各檔案撰寫要求已詳述於上方表格。

本題將使用以下的 Makefile 進行編譯：

```
all: main.cpp operator.o
    c++ main.cpp operator.o
operator.o: operator.cpp operator.h
    c++ operator.cpp -c
clean:
    rm -f *.o *~ *.*~ a.out*
```

相關測試檔路徑：

1. /home/stu/public/cpp2026s/cpp.hw11/p3/in.1
2. /home/stu/public/cpp2026s/cpp.hw11/p3/in.2
3. /home/stu/public/cpp2026s/cpp.hw11/p3/in.3

此題的執行結果可參考如下：

```
[3:23▲user@ws▲p3]▲./a.out▲<▲in.1↵
Paragraph:▲↵
The▲world▲of▲gaming▲changed▲forever▲in▲1994▲when▲Sony▲introduced▲the▲original▲PlayStation.▲As▲a▲groundbreaking▲home▲video▲game▲console,▲the▲first▲PlayStation▲revolutionized▲the▲industry▲by▲popularizing▲3D▲graphics▲and▲CD-ROM▲technology.▲As▲the▲years▲progressed,▲the▲brand▲evolved▲into▲a▲household▲name.▲The▲PlayStation▲2▲shattered▲records▲to▲become▲the▲best-selling▲console▲of▲all▲time,▲followed▲by▲the▲connected▲ecosystem▲of▲the▲PlayStation▲3▲and▲the▲massive▲community▲built▲around▲the▲PlayStation▲4.▲Today,▲the▲PlayStation▲5▲pushes▲the▲boundaries▲of▲modern▲gaming▲with▲lightning-fast▲SSD▲speeds,▲immersive▲haptic▲feedback,▲and▲breathtaking▲visuals.↵
↵
Needle:▲[PlayStation]↵
↵
Occurrences▲of▲needle:▲6↵
[3:23▲user@ws▲p3]▲./a.out▲<▲in.2↵
Paragraph:▲↵
The▲world▲of▲gaming▲changed▲forever▲in▲1994▲when▲Sony▲introduced▲the▲original▲PlayStation.▲As▲a▲groundbreaking▲home▲video▲game▲console,▲the▲first▲PlayStation▲revolutionized▲the▲industry▲by▲popularizing▲3D▲graphics▲and▲CD-ROM▲technology.▲As▲the▲years▲progressed,▲the▲brand▲evolved▲into▲a▲household▲name.▲The▲PlayStation▲2▲shattered▲records▲to▲become▲the▲best-
```

```
selling▲console▲of▲all▲time,▲followed▲by▲the▲connected▲ecosystem▲of▲the▲PlayStation▲3▲and▲th
e▲massive▲community▲built▲around▲the▲PlayStation▲4.▲Today,▲the▲PlayStation▲5▲pushes▲the▲bo
undaries▲of▲modern▲gaming▲with▲lightning-
fast▲SSD▲speeds,▲immersive▲haptic▲feedback,▲and▲breathtaking▲visuals.↵
↵
Needle:▲[,]↵
↵
Occurrences▲of▲needle:▲6↵
[3:23▲user@ws▲p3]▲./a.out▲<▲in.3↵
Paragraph:▲↵
AAAAAA↵
↵
Needle:▲[AAAA]↵
↵
Occurrences▲of▲needle:▲3↵
[3:23▲user@ws▲p3]
```

- 本題相關的程式碼路徑已註明於檔名右側，同學們可以透過路徑複製到自己的家目錄。
- 類別撰寫要求請參考上方表格
- 本題應繳交檔案如下：
  - operator.h
  - operator.cpp

p4 員工與任務

在鳳梨資訊公司中，看板（**カンバン**Kanban）常被來追蹤員工當前的工作。

請參考下列的 main.cpp 程式（請依據檔名旁的路徑至 ws 取得程式碼）：

```
#include <iostream>
#include <string>
using namespace std;

class Task
{
protected:
    string name;

public:
    void set_name(string n);
    string get_name();
};

class AssignTaskTable
```

```
{
protected:
    Task task[5];
    int task_count = 0;

public:
    void operator=(const AssignTaskTable &task_table); // overwrite all
tasks with task_table
    void operator=(Task &t);                           // overwrite all
tasks with t
    void operator+=(Task &t);                          // add a new task
    void operator-=(Task &t);                          // remove a task
(task name shoule be checked)
    AssignTaskTable operator+(AssignTaskTable &others_task_table);

    int get_task_count();
    Task *get_task_at(int index);
};

class Employee
{
public:
    string name;
    AssignTaskTable assigned_task;
    void set_name(string n);
    string get_name();
};

ostream &operator<<(ostream &out_string, Employee &e);
```

請參考下列的 main.cpp 程式（請依據檔名旁的路徑至 ws 取得程式碼）：

```
#include "kanban.h"

int main()
{
    Employee Amy, Bob, Tony;
    Task task[5];

    task[0].set_name("View website request log");
    task[1].set_name("Edit new version cybersecurity playbook");
    task[2].set_name("Edit API documents");
    task[3].set_name("Data cleaning for database");
    task[4].set_name("Monitoring request data: application traffic");

    Bob.set_name("Bob");
    Amy.set_name("Amy");
```

```
    Tony.set_name("Tony");

    Bob.assigned_task = task[0];
    cout << Bob;
    Bob.assigned_task += task[1];
    cout << Bob;
    Bob.assigned_task += task[2];
    cout << Bob;
    Bob.assigned_task = task[2];
    cout << Bob;
    Bob.assigned_task += task[3];
    cout << Bob;

    cout << "--------------" << endl;

    Amy.assigned_task = task[3];
    cout << Amy;
    Amy.assigned_task += task[4];
    cout << Amy;

    cout << "--------------" << endl;

    Tony.assigned_task = Bob.assigned_task + Amy.assigned_task;
    cout << Tony;
    Tony.assigned_task -= task[0]; // invalid operation: do nothing
    cout << Tony;
    Tony.assigned_task -= task[4];
    cout << Tony;

    return 0;
}
```

在此題中，將運算子多載融合進鳳梨資訊公司。在這裡，每位「員工（Employee）」都是一個獨立的物件，而他們內部的「任務看板/任務清單（AssignTaskTable）」則負責維護員工當前的所有工作狀態。

我們賦予「=」一個新的定義：「覆蓋指派運算子」。將一個特定的任務賦予員工時，代表公司希望該員工專注。此時，員工原先擁有的所有任務都會全數移除，改為專心完成這項新指派的工作（main.cpp line 18, 24, 31）。此外，若將另一位同事的任務清單整個指派給該員工，則代表該員工將共同執行任務清單上的內容（main.cpp line 36）。

我們賦予「+=」一個新的定義：「追加任務運算子」。指派一個新任務給該員工，新指派的任務並不會影響舊有的工作，而是會被依序加入到該員工任務列表的末端，不過每位員工的任務清單上限最多只能同時容納五個任務（題目會確保增加的任務在 5 個以內，無需排錯）。

我們賦予「-=」一個新的定義：「解除指派運算子」。當某項工作完成或需要移除時，我們會透過此運算子變更員工的任務清單，比對員工看板內所有的工作名稱，當發現與欲刪除任務的名稱完全相同，就會將其從列表中移除；若完全找不到相符的任務，員工的狀態則保持不變。

有時候專案需要多人協作，我們賦予「+」一個新的定義：「合併運算子」，來進行任務的交接與整合。此操作能將兩位員工各自的任務清單進行合併，並產出一個包含雙方所有工作內容（同名的任務只會存在一個）的清單。此外，參考 main.cpp 第 38 行，可透過我們先前定義的「=」運算子，將整份清單指派給第三位員工。

最後，當我們想知道員工目前任務清單的內容，可以透過串流輸出運算子（<<）顯示該員工任務清單的內容。若目前沒有任何工作，方括號內部則會保持留空。

檔案名稱	**檔案用途與撰寫要求**	**-**
main.cpp	主程式	題目提供原始碼
kanban.h	Task、AssignTaskTable 與 Employee 類別的定義	題目提供原始碼
kanban.cpp	Task、AssignTaskTable 與 Employee 類別的實作	需繳交

請完成名為 **kanban.cpp** 的 C++ 語言程式，各檔案撰寫要求已詳述於上方表格。

本題將使用以下的 Makefile 進行編譯：

```
all: main.cpp kanban.o
    c++ main.cpp kanban.o
kanban.o: kanban.cpp kanban.h
    c++ kanban.cpp -c
clean:
    rm -f *.o *~ *.*~ a.out*
```

此題的執行結果可參考如下：

```
[3:23 user@ws p4]./a.out
Bob is currently assigned to [View website request log]
Bob is currently assigned to [View website request log, Edit new version cybersecurity playbook]
Bob is currently assigned to [View website request log, Edit new version cybersecurity playbook, Edit API documents]
Bob is currently assigned to [Edit API documents]
Bob is currently assigned to [Edit API documents, Data cleaning for database]
--------------
Amy is currently assigned to [Data cleaning for database]
Amy is currently assigned to [Data cleaning for database, Monitoring request data: application traffic]
--------------
Tony is currently assigned to [Edit API documents, Data cleaning for database, Monitoring request data: application traffic]
Tony is currently assigned to [Edit API documents, Data cleaning for database, Monitoring request data: application traffic]
Tony is currently assigned to [Edit API documents, Data cleaning for database]
[3:23 user@ws p4]
```

- 本題相關的程式碼路徑已註明於檔名右側，同學們可以透過路徑複製到自己的家目錄。
- 類別撰寫要求請參考上方表格。

- 本題應繳交檔案如下：
  - kanban.cpp