《程式語言》

試題評析	今年除了第四題外，均爲老師上課提醒的重點項目，包括物件導向的概念、抽象資料結構以及遞迴程式撰寫.第一題是物件導向繼承的概念；第二題是控制結構章節中流程圖的畫法；第三題是遞迴程式的撰寫；第四題則比較需要實務軟體開發經驗才能回答完整；第五題是抽象資料結構的程式撰寫。預估一般分數落在 60~70 分，具有實務開發經驗以及對物件導向程式設計概念熟悉的同學可以拿到 80 分以上。

一、(一)試說明繼承(inheritance)在物件導向程式語言(object-oriented programming languages)中的角色及重要性。(10 分)

(二)試比較 C++的多重繼承（multiple inheritance）和 Java 的 interfaces。(10 分)

【擬答】

(一)

繼承的角色：

繼承是物件導向程式語言中所產生新類別的一種方式，可以定一個新的類別(稱子類別)，並且使此子類別繼承另一類別(稱父類別)，則子類別會擁有父類別的行爲和屬性，子類別也可以重新定義父類別已經定義過的方法或增加專屬於子類別的方法或屬性。

繼承的重要性：

1.繼承的機制可以節省程式碼的撰寫：由於子類別可以繼承父類別的方法及屬性，不用重新定義一次，所以可以節省程式碼撰寫。

2.繼承和動態繫結是用來實做物件導向中多型概念的方法，利用繼承可以使多個子類別繼承自同一個父類別，將此父類別視爲這些子類別對外相同的溝通管道，亦即外界的物件想和這些子類別的物件溝通時，必須透過父類別這個介面。

(二)

1.C++允許一個類別繼承自多個不同的父類別，而 Java 的類別只能繼承單一父類別，但可以實作多個介面，且介面本身可以繼承自多個介面，以此來達成多重繼承的效果，但不能去定義介面中的方法。

2.C++多重繼承的功能較廣，由於 Java 的介面只能定義方法的宣告，不能定義方法的實作或是物件變數，而 C++的父類別則無此限制，且 C++類別中的方法可以宣告成虛擬函式，也可以是一般類別成員函式，但 Java 類別或介面中的所有方法均爲虛擬函式，所以 Java interface 能達成的功能是 C++多重繼承的子集合。

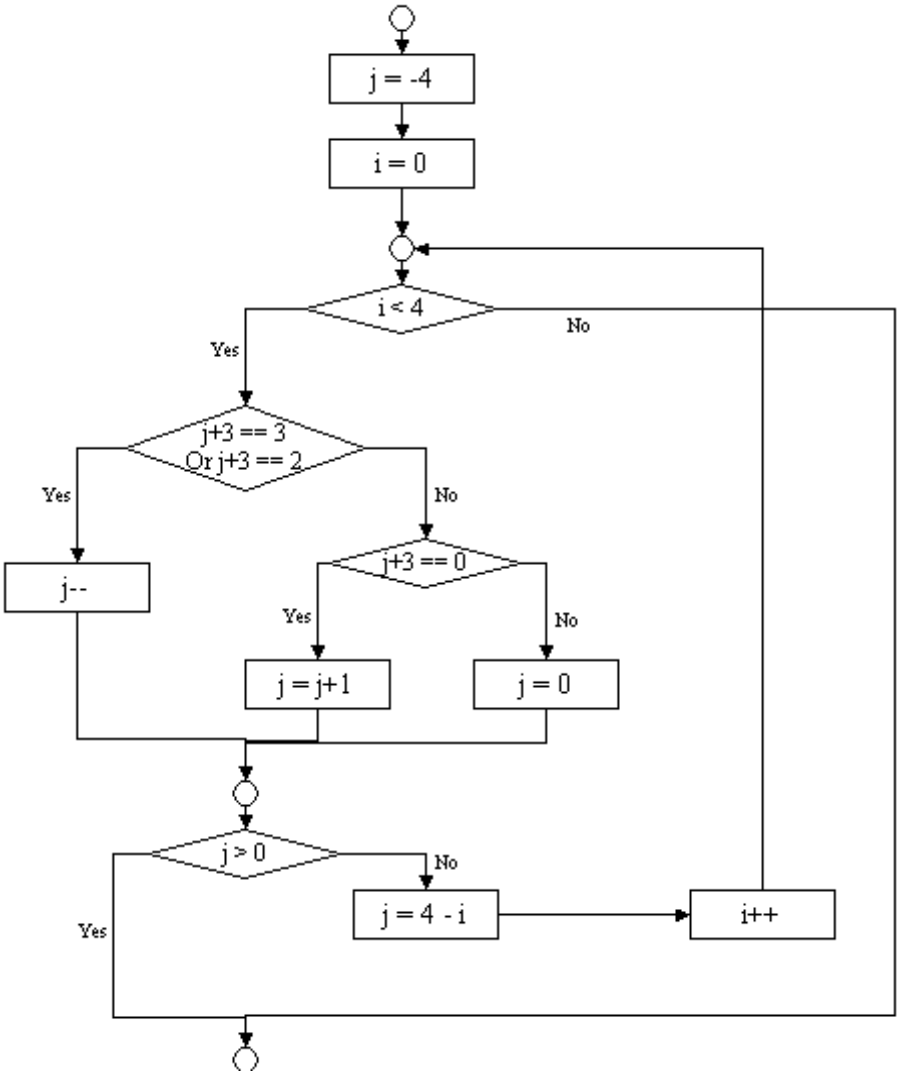
3.C++的多重繼承可以達成程式碼重複使用，亦即不用重新定義父類別中定義過的函式，但是 Java 的介面不能去定義方法，而只能宣告方法來達成多型的目的，不能達成重複使用程式碼的效果。

4.C++是以 class A ， public B， public C 的語法來達成類別 A 多重繼承類別 B 及 C，Java 的繼承是以 extends 關鍵字來達成，而實作介面是以 implements 關鍵字達成，class A extends B implements C 則表示類別 A 繼承類別 B 且實作介面 C。

二、考慮下列 C 程式片段，請畫出其流程圖。(20 分)

```
j=-4;
for(i=0;i<4;i++){
    switch (j+3){
       case 3：
       case 2：j--;break;
       case0：j=j+1;break;
       default：j=0;
    }
    if(j>0)break;
    j=4-i;
}
```

【擬答】



三、請以任何一程式語言，分別以遞迴（recursive）及非遞迴演算法來計算利用 N 條線將平面分成最多的區域，其中 N 為大於等於 0 之整數。（20 分）

【擬答】

(1)遞迴解法

```
int planeNum1(int num)
{
	if(0 >= num) {

		return 1;
	} else {

		return planeNum1(num - 1) + num;
	}
}
```

(2)迴圈解法

```
int planeNum2(int num)
{
	int previousResult = 1;
	int currentResult = previousResult;
	if (num > 0) {

	    for (int i = 1; i <= num; ++i){

		currentResult = previousResult + i;
		previousResult = currentResult;
	    }
	} else {0
	    ;
	}
	return currentResult;
}


void main()
{
	int lineNum = 0;
	for (int i = 0; i < 5; ++i) {

		cout << "輸入直線數目(>=0)-->";
		cin >> lineNum;
		cout << "1.可分割成 " << planeNum1(lineNum) << " 個平面" << endl;
		cout << "2.可分割成 " << planeNum2(lineNum) << " 個平面" << endl;
		cout << endl << endl;
	}
}
```

四、系統委外開發越來越普遍，試從程式語言觀點說明程式開發委外其可能問題（含軟體安全）及解決方式。（10 分）

【擬答】

(一)需求制定的問題：

1.制定的需求在技術上無法達成

解決方式：

(1)探討所選擇的語言工具是否不適合開發此系統，不同的語言會有適合開發的特定領域系統。

(2)探討需求是否必要。

(3)是否有其他功能的組合可以達成所需的功能。

(二)開發過程中的問題：

1.需求增加或變更

解決方式：

(1)在需求制定階段就探討未來系統變更的可能性，讓軟體架構設計可以考慮未來的可能變動，並且需求有變動時以書面提出，與廠商共同溝通解決，留下明確紀錄。

(2)選擇可以設計較佳彈性的程式語言，例如：物件導向程式語言。

2.系統開發進度落後

解決方式：

(1)設立檢查點，請廠商在檢查點提供相關進度資訊。

(三)系統開發完成後的問題：

1.軟體安全性不佳或系統不穩定

解決方式：在驗收時已使用大量資料去測試此系統，分別針對系統的一般操作，安全性來設計測試資料。

2.系統和內部軟體或硬體平台環境不相容

解決方式：在需求制定階段就和廠商溝通清楚所需要的內部環境設定，並且可以於檢查點讓委外系統在內部做測試。

3.系統效能不佳

解決方式：和委外廠商定義出合理的效能範圍，利用開發的程式語言以及所使用的環境來最佳化所開發的系統。

(四)系統維護的問題：

1.功能上需要增加

解決方式：所使用的軟體開發工具或語言能夠應付未來需求的變化，例如：以物件導向程式語言開發系統。

2.系統穩定的品質

解決方式：建立迴歸測試環境(regression)，建立一份測試資料集來測試已經完成且正確的現有功能，未來任何的系統改變必須先通過這份測試資料，才能確保系統品質穩定。

五、請以任一種程式語言（或虛擬語法）寫出一資料抽象型態（data abstract）之 Stack 結構（先進後出），另至少必須包含有 initialization， insertion，及 deletion 等運算。（20 分）

【擬答】

```
#define TOTAL_DATA_NUM 10
class stack {
public:
	stack();
	void push(int data);
	int pop();
private:
	int m_nTotalData[TOTAL_DATA_NUM];
	int m_nElementCount;
};
stack::stack()
{
	m_nElementCount = 0;
}
void stack::push(int data)
{
	if (m_nElementCount == TOTAL_DATA_NUM) {
	 cout << “Stack is full” << endl;
	} else {
	 m_nTotalData[m_nElementCount++] = data;
	}
}

int stack::pop()
{
	if (m_nElementCount == 0) {
	 cout << “Stack is empty” << endl;
	 return -1;
	} else {
	 return m_nTotalData[--m_nElementCount];
	}
}
```