第一題

(a) Size_t 是自訂型別，一種用來記錄大小的資料類型，不使用的話比較不容易記住宣告的變數該儲存什麼。

(b) EOF 是指一個文字檔的結尾，讀檔時可利用 EOF 判斷是否到文字檔的結尾。

(c)

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

int main ()
{
	int money_num = 0;
	cin >> money_num;
	string money = to_string(money_num);		//將 int 轉為 string
	money.insert(0,"$");				//最前面先加$
	for(int i = 0; i < money.size() ;i++)		//依據幾個數字來跑回圈
	{
		if(i%4 == 3 && money.size() - i != 1)	//每跑三個就要加一個逗點，且$後面不能加逗點
			money.insert(money.size() - i, ",");	//從後面加逗號
	}
	cout << money;
	return 0;
}
```

第二題

(a)當創建一個 class 時，不需要重新編寫新的資料成員和成員函數，只需指定新建的 class 繼承了一個已有的 class 的成員即可，繼承代表了 B is A 的關係。某個 class B 可以繼承另外一個 class A，class B 就可以存取 class A 的成員，而繼承有分三種類別：

公有繼承（public）：當 class B 以公有繼承 class A，class A 的公開成員也是 class B 的公開成員，class A 的保護成員也是 class B 的保護成員。

保護繼承（protected）：當 class B 以保護繼承 class A，class A 的公有和保護成員將成為 class B 的保護成員。

私有繼承（private）：當 class B 以私有繼承 class A，class A 的公有和保護成員將成為 class B 的私有成員。

以本次作業第三題為例：船的資訊(船舶編號、船舶重量、國籍、船長姓名、是否裝在危險物品)、工作的資訊(該工作船舶的編號、可出發時間、類型、港口、起點瑪頭、終點瑪頭)，並依給定的演算法算出正確答案。

很明顯的我們會有兩個 class 其一為船，船的屬性包括船舶編號、船舶重量、國籍、船長姓名、是否裝在危險物品以及 constructer 等必要函數(因題目給定之演算法不會用到船的資訊，因此無 getxxx 函數)；其二為工作，工作的屬性包括該工作船舶的編號、可出發時間、類型、港口以及 constructer、getxxx 等必要函數。

再來會依照工作類別繼承工作產生三個 class：移泊，屬性包括起點瑪頭、終點瑪頭；進港，屬性包括港口、終點瑪頭；出港，屬性包括港口、起點瑪頭。而因為工作與船的數量都大於等於 1，因此再寫兩個 class：船陣列、工作陣列，是必須的。因三種類型的工作的屬性不一，因此若不用繼承產生三個子 class 便會有空的屬性，而這次的演算只會用到出港工作以及進港工作，因此分三種工作來討論有助於寫演算法，若沒有這樣繼承便要不停的判斷該工作為哪種工作，容易發生 logic error。

(b)動態繫結是指在執行時期才會得知實際呼叫哪個函式，多型可讓我們做到動態繫結。

一個父 class 的物件指標，可以用來指向其子 class 而不會發生錯誤，因為這個原因，當我們使用父類別的指標當作函式的參數，實際呼叫函式時，可以傳入指向父類別或子類別的指標。

因為是父 class 的指標，因此只能呼叫父 class 的函數，若想要呼叫子 class 的函數，需要在父 class 把要多型的函式前加上關鍵字 virtual，表示這函式是虛擬函式，並在子 class 中重新定義虛擬函式，表示此成員函式的操作延遲至執行時期再決定。

承上題，使用多型便可讓我們將三種工作皆儲存在工作 class 的指標陣列中，而該題演算法須依港口分別寫，因此我們需要知道該工作使用之港口為何，但是只有子 class 有港口，這時 virtual 函數便派上用場。如果不使用這樣的多型，我們會需要開三條陣列來分別儲存三種工作，會使一些想法比較沒有那麼直覺