參考下列程式碼中，依序回答下列問題：

(1) #2, #3 兩行程式碼的意義。

(2) 若 #3 沒有寫的話，compile 會發生錯誤，請問為什麼？以及有哪些解決的方法呢？

(3) #1, #2 有兩種 include 的方法（意指 ”xxx” 及 <xxx> 兩種），請問這兩種方法有何差異。

#include "stdafx.h" // #1

#include <iostream> // #2

using namespace std; // #3

int \_tmain(int argc, \_TCHAR\* argv[])

{

cout << "Hello world!";

return 0;

}

(1)   
#include <iostream>   
是一個預處理指令，它可以把標頭檔<iostream>插入程式源文件中該指令的所在位置。該#include的動作是在程式編譯前完成的，使程式可以使用<iostream>的內容。

using namespace std;  
這個指令是指從源文件的這個地方開始可以使用名稱空間std中的名稱，若沒有這行指令，編譯器不會知道cout的定義。

(2)

如果沒有寫(3)的話，編譯器會無法識別cout << "Hello world!";這行指令的意義，因為cout是被定義在std裡面的。

解決方法：可以在每次用到cout時寫成std::cout，例如此例可改成std::cout << "Hello world!";, 或者可以在#3的位置寫上using std::cout;來使用cout指令。

(3)

"xxx"及<xxx>兩種include的方法主要區別是查找該程式的過程。這個操作會隨著編譯器而有所不同。一般情況下，使用<xxx>時，編譯器只會搜尋包含標準庫標頭檔中的默認目錄。若是使用"xxx"，編譯器就會搜索當前目錄（一般是正在編譯的原始檔案所在的目錄），之後搜索包括標準標頭檔的目錄。如果我們把標頭檔放在其他目錄下，為了查找到它，必須在雙引號中指定從原始檔案到標頭檔的完整路徑。在一些C++的實現方式中，可以指定包含原始檔案目錄的相對路徑。