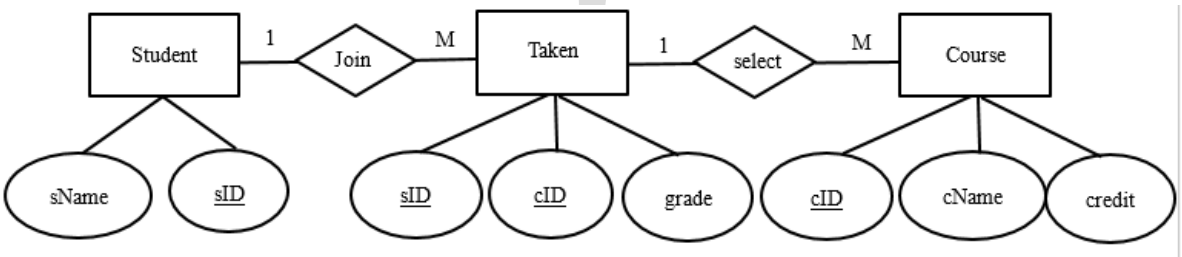
104 年特種考試地方政府公務人員考試試題

等 別：三等考試
類 科：統計
科 目：資料處理

一、電腦的記憶體分為那幾種？請比較它們的功能、速度和大小。
【擬答】：
記憶體可分為 RAM 跟 ROM 兩大類
RAM：Random Access Memory，中文稱為隨機存取記憶體，是一種可讀寫的記憶體，用來暫時保存資料，速度比 ROM 快，通常作為作業系統與其他行程的臨時資料存儲媒介。儲存在 RAM 中的資料會隨著系統關機或是程式結束而消失。當作業系統在執行運算時，會先將儲存在硬碟的資料載入到 RAM，再把 RAM 的資料存至 CPU 的暫存器來運算。需這樣是因為這些儲存設備間的速度落差極大，如此可提升執行的效率。
ROM：Read-Only Memory，中文稱為唯讀記憶體，特性是儲存的資料無法改變或刪除，內容也不會因為電源關閉而消失。ROM 通常用來儲存不需經常變更的程式或資料，例如電腦的 BIOS。下表是 RAM 與 ROM 的差別：

元件	RAM	ROM
速度	快	慢
存取特性	可讀寫	唯讀
儲存時間	揮發性記憶體	非揮發性記憶體
用途	存放執行中的程式及資料	存放開機系統程式
資料特性	電源開啟時可存取，無法永久保存	永久存在

二、考慮關聯式資料庫的三個 table：學生（Student）、課程（Course）和修課紀錄（Taken）。它們的結構（schema）定義如下：Student（sName, sID）、Course（cName, cID, credit）、Taken（sID, cID, grade）。
【擬答】：
㈠以 E-R diagram 畫出三個 table 的關係，並標註 table 中的 primary key 和 foreign key。



㈡給定學生姓名（王小明），請以 SQL 語法列出學生姓名（sName）、學號（sID）和學生的加權平均成績（GPA）。一個學生的一門課的加權成績是 Course.credit*Taken.grade，一個學生的總加權成績是所有修過課加權成績的總和，一個學生的 GPA 計算如下：

$$GPA=\frac{\text{學生修課的總加權成績}}{\text{學生總修課學分數}}$$

注意可能會有學生有相同的姓名，查詢必須列出所有相同姓名學生的 GPA，並且以學生的 sID 大小排列。

```
Select Student.sName, Student.sID,
(sum(Course.credit*Taken.grade)/sum(Course.credit)) as GPA
```

```
from Student,Course,Taken
where Taken.cID=Course.cID
group by sID
order by sID asc
```

三、請回答下列關於 IP 網路封包欄位的問題：

㈠ TTL 的作用是什麼？請以 IP routing 的原理說明為何需要 TTL 這個欄位。

㈡ header checksum 的作用是什麼？為何 checksum 可以達到這個功能？

【擬答】：

TTL 是 Time To Live 的縮寫，用來表示 IP 封包被路由器丟棄前允許通過的網段數量。在 IP 協定中，TTL 是以 hop 為單位。封包每經過一個 router，此值就會自動減一。當封包的 TTL 值等於 0 的時候，該封包就會被丟棄。這樣可以避免封包在傳遞過程中因為某些因素未能抵達目的地，造成該封包一直在網路上傳送。

header checksum 是負責檢查 IP 表頭有沒有問題的一個欄位。若檢驗數值不合，此封包就會被丟棄。當資料要傳送出去時，發送端會先對資料進行校驗，再將校驗值放至 header checksum。接收端收到封包後，會對資料進行校驗，再比對校驗值是否一致。若結果不一致則會將該資料視為已損毀，要求發送端方重送。

四、一個 binary tree 的 node 定義如下：

```
struct node {
  int value;
  struct node *left,
*right;
};
```

寫出一個 recursive function，int maxvalue (struct node *p) ，找出一個 binary tree 中的最大值。

【擬答】：

```
int maxvalue (struct node *p){
   int root, left, right, max = -1;

   if(p){
      root = root->data;
      left = maxvalue (root->left);
      right = maxvalue (root->right);

      if(left > right)
          max = left;
      else
          max = right;

      if(root > max)
          max = root;
   }
   return max;
}
```