資料結構 HW4 資訊工程二甲 陳俊宇 F64051114

此次作業，是利用XOR特性來精簡雙向鏈結成單向鏈結，而由左至右與由右至左我運用了兩個不同函數，分別是creat\_head\_trail\_linked\_list函數(設置由左至右)與insert函數(設置由右至左)。

其實，creat\_head\_trail\_linked\_list函數也可以做由右至左，因為我把整個list的頭跟尾位置都記錄下來了，取用的起始位置可以被我靈活運用。而我是先想出insert的方法才發現這樣無法做由左至右，故再多做了一個creat\_head\_trail\_linked\_list函數。insert函數，他的自由度不高，因為他的head指標最後只會動到整串list的尾巴，若追蹤list，肯定只能由右至左；而creat\_head\_trail\_linked\_list函數我是在data等於1時，凍結住head指標的位置，而trail指標跟著lastnode移動，最後就會停止在data of node為20處。

我覺得此次作業讓我最頭痛處，在於XOR的靈活運用，他是把鏈結產生關係的因素，像是我程式內第71行與56行與91行與28行，他們都是相同的概念，我這邊解釋第28行：nodePointer left = XOR(lastnode->link, NULL)，此if敘述除了while第一次迴圈外，都會執行。而它的意義就是我要知道lastnode的link值，就是要從它左邊的node(left)與右邊的node(newnode)做XOR運算，而左邊的node(left)我們透過的特性做代換，怎麼說呢？未有newnode 之前,lastnode->link=left^NULL,再XOR一個NULL來產生left。這便是XOR的運用法。