HW4: Pointer Lecture Reading

在 CS 50 Software Design and Implementation Lecture 8 Pesky Pointer 中除了提到指標在 C 語言中的特性，還將 C 語言和 Java 的 Pointer(實際上為 references)做了一些對比。在 Java 中使用 References 為一唯一用法幾乎不會出錯並且有嚴謹的規則，然而在 C 中有常見的標準用法對 Value 直接進行改動，還有另一種則是將 Address 的 Value 改動，這兩種不同的用法直接造成了課堂上提到的 Swap Trap。原理是原本直接調整變數裡的值，如果用 Pointer 指到該變數進行操作，那擁有該 Address 的 Variable 或是值為於該 Address 的 Variable 也會受到影響。

```
#include <stdio.h>
void swap(int *ip, int *jp) {

    int temp;
    temp = *ip;
    *ip = *jp;
    *jp = temp;

}

int main(int argc, char *argv[]) {

  int a=3, b=5;
  printf("before a= %d, b= %d\n",a,b);
  swap(&a, &b);
  printf("after a= %d, b= %d\n",a,b);
  return(0);

}
```

如上圖，當 a、b 的 Value 傳入 swap function 中，該 Function 直接對二變數的 Address 進行操作使 Function swap 後不用 return a, b's value 直接被影響到。

Conclusion: With using of Pointer, we can not only inference the value in address of variables, but also return a lot of increasing, decreasing, swapping, and shifting the value of variables and easily to dealing with the return of value.

第二段則是解釋 pointer 和 array 之間的關係還有&和->，如下圖所示

assignment statements are the same, and the first

```
char buffer[BUFSIZ], *ptr;
ptr = buffer;
ptr = &buffer[0];
```

-> explain char *ptr

Ptr = buffer; 和 ptr = &buffer[0], 在 ptr 拿到 buffer 的 address 沒有加減 array 位置的情況下，兩者意義是一樣的皆為 ptr 一個只到 char 的 pointer 皆會拿到 buffer 位於[0]的 address。->則是用在 structure 的 pointer。

比較有意思的是 strcpy 在 while loop 時先++和後++會產生巨大的差異，後++的執行到 len13 會錯誤但還會+1，但先++的會先+1 偵測到 13 是'\0'後就會停止這時 function 就會正確