請說明下列程式碼片段可能有什麼樣的問題？該如何修正？（只要建議修正的方法就好，不用寫修正過後的程式片段）

1.

```
enum { TABSIZE = 100 };

static int table[TABSIZE];

int *f(int index) {
  if (index < TABSIZE) {
    return table + index;
  }
  return NULL;
}
```

A :

* 若 **index** 小於 **0**，會用到超過陣列範圍的記憶體，導致程式不安全。
* 解決方法 : 在 **if** 的判斷條件中，加入 **index>=0** 的判斷。

2.

```
#define BUFSIZE 256
int main(int argc, char **argv) {
char *buf1 = (char *) malloc(BUFSIZE);
char *buf2 = (char *) malloc(BUFSIZE);
strcpy(buf1, argv[1]);
free(buf2);
}
```

A :

* 這個程式直接用 **strcpy**，把 **argv[1]**的字串塞入 **buf1**。然而並沒有先檢查使用者的輸入，輸入可能會超過 **BUFFER SIZE256**。這可能會有 **buffer overflow** 的情形產生，導致程式不安全。
* 解決方法 : 對 **argv[1]**這個由使用者讀入的字串，先進行長度的檢查，確定不會超過 **buffer** 的大小之後，再使用 **strcpy** 比較合適。或是使用 **strncpy**，直接對讀入字串有限制。

3.

```
unsigned int readdata () {
int amount = 0;
...
if (result == ERROR)
amount = -1;
...
return amount;
}
```

A

*** 這個函式原本定義回傳 UNSIGNED INT**
但是，回傳了一個 INT 的 amount，這會超出 unsigned int 所包含的範圍。
*** 解決方法 ： 把定義函式的回傳 type 改成 int。**

4.

```
char* ptr = (char*)malloc (SIZE);
if (err) {
abrt = 1;
free(ptr);
}
...
if (abrt) {
logError("operation aborted before commit", ptr);
}
```

A ：
*** 若是上面判斷式中，err 的條見滿足了。ptr 會被 free 掉，此時若 abrt 的條件也滿足，則會使用到已經 free 掉的記憶體空間，導致程式有安全上的問題。**
*** 解決方法 ： 在 abrt 的條件判斷中，多加上對於 err 的判斷。 if(abrt && !err) ，要滿足這個條件，才可以使用 ptr 的指標空間。**

5.

```
#include <stdio.h>

void printWrapper(char *string) {

printf(string);
}

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

char buf[5012];
memcpy(buf, argv[1], 5012);
printWrapper(argv[1]);
return (0);
}
```

A :

*** 這段程式碼的 printwrapper 會呼叫 printf，直接把字串印出來。這真的是很可怕的一件事情，因為有心人士可以藉這個機會，進行 format string attack。例如 : 偷偷輸入"%p"，印出程式中的記憶體的位置資訊，若讓有心人士得到這個資訊，就會讓這個程式有安全上的問題。**

*** 解決方式 : 和 Q2 的方法很像，就是要對使用者的讀入資料先做檢查，如果使用者故意輸入像是"%p"這種資料，就不可以讓他印出東西。**