Homework 1

Below are four faulty programs. Each includes test inputs that result in failure. Answer the following questions about each program.

```
'''
Find last index of element

@param x array to search
@param y value to look for
@return last index of y in x; -1 if absent
@TypeError if x is None or …
'''
def find_last(x, y):
    i = len(x) - 1
    while i > 0:
        if x[i] == y:
            return i
        i -= 1
    return -1
# test: x = [2, 3, 5]; y = 2; Expected = 0
```

```
'''
Find last index of zero

@param x array to search

@return last index of 0 in x; -1 if absent
@TypeError if x is None or …
'''
def last_zero(x):
    i = 0
    while i < len(x):
        if x[i] == 0:
            return i
        i += 1
    return -1
# test: x = [0, 1, 0]; Expected = 2
```

```
'''
Count positive elements

@param x array to search
@return count of positive elements in x
@ TypeError if x is None or …
'''
def count_positive(x):
    count = 0
    i = 0
    while i < len(x):
        if x[i] >= 0:
            count += 1
        i += 1
    return count
# test: x = [-4, 2, 0, 2]; Expcted = 2
```

```
'''
Count odd or postive elements

@param x array to search
@return count of odd/positive values in x
@ TypeError if x is None or …
'''
def odd_or_pos(x):
    count = 0
    i = 0
    while i < len(x):
        if x[i] > 0 or x[i] % 2 == -1:
            count += 1
        i += 1
    return count
# test: x = [-3, -2, 0, 1, 4]; Expected = 3
```

(a) Explain what is wrong with the given code. Describe the fault precisely by proposing a modification to the code.

ANS:

find_last(x, y):

1. while i > 0 要改成 i >= 0

因為 index 0 沒有被涵蓋到，index 最小是 0。

last_zero(x):

1. i = 0 要改成 i = len(x) – 1
2. while i < len(x) 要改成 while i >= 0
3. i += 1 要改成 i -= 1

因為要從後面開始搜尋才能找到最後一個 0 的 index，原本的程式從前面找到第一個就會 return 了。

count_positive(x):

1. if x[i] >= 0 要改成 if x[i] > 0

因為 0 不屬於正整數

odd_or_pos(x):

1. x[i] % 2 == -1 要改成 x[i] % 2 == 1

因為負數的餘數是正數，改成用==1 來判斷。

(b) If possible, give a test case that does not execute the fault. If not, briefly explain why not. (You need to give the same number of arguments.)

ANS:
find_last(x, y):
x = None; y = None; Expected = TypeError; actual = TypeError
會直接回傳 typeError，不會觸發 fault 的程式碼

last_zero(x):
x = None; Expected = TypeError; actual = TypeError
會直接回傳 typeError，不會觸發 fault 的程式碼。

count_positive(x):
x = []; Expected = -1; actual = -1
只要 len(x) = 0 就不會執行 while 迴圈，不會觸發有 fault 的 if 判斷句(if x[i] >= 0 要改成 if x[i] > 0)。
或是，
x = None; Expected = TypeError; actual = TypeError
會直接回傳 typeError。

odd_or_pos(x):
x = []; Expected = -1; actual = -1
只要 len(x) = 0 就不會執行 while 迴圈，不會觸發有 fault 的 if 判斷句(x[i] % 2 == -1 要改成 x[i] % 2 == 1)。
或是，
x = None; Expected = TypeError; actual = TypeError
會直接回傳 typeError。

(c) If possible, give a test case that executes the fault, but does not result in an error state. If not, briefly explain why not. (You also need to answer expected and actual output.)

ANS:
find_last(x, y):
x = [0, 1, 2, 3]; y = 3; Expected = 3; actual = 3
只要 y 要找的目標不在 index 為 0 的位置，在觸發未檢查 index 0 的情況前就已經 return 了，不會發生 error state(未檢查 index 0)。

last_zero(x):
x = [3, 2, 1, 0]; Expected = 3; actual = 3
只要 0 只有一個，就不會發生 error state(找到第一個 0 就 return)。

count_positive(x):
x = [-4, 2, 1, 2]; Expected = 3; actual = 3
只要不出現 0，就可以計算出正確的正整數的數量，不會發生 error state(把 0 計算成正數)

odd_or_pos(x):
x = [0, 1, 2, 4]; Expected = 2; actual = 2
只要不出現負數，就可以計算出正確的數量，不會發生 error state(無法正確計算 negative odd number)。

(d) If possible, give a test case that results in an error state, but not a failure. If not, briefly explain why not. (You also need to answer expected and actual output.)

ANS:

find_last(x, y):

x = [0, 1, 2, 3]; y = 4; Expected = -1; actual = -1

y 要找的目標不在 x 中，會觸發未檢查 index 0 的情況，但不會 failure。

last_zero(x):

x = [3, 2, 1]; Expected = -1; actual = -1

只要沒有 0，會發生 error state，但不會 failure。

count_positive(x):

辦不到，發生 error state 就一定導致 failure。

。

odd_or_pos(x):

辦不到，發生 error state 就一定導致 failure。

(e) For the given test case in (d), describe the first error state. Be sure to describe the complete state.

ANS:

find_last(x, y):

x = [0, 1, 2, 3]; y = 4; Expected = -1; actual = -1

First error state:

x = [0, 1, 2, 3]

y = 4

i = 0

執行到迴圈結束，在執行 return -1 之前。

last_zero(x):

x = [3, 2, 1]; Expected = -1; actual = -1

First error state:

x = [3, 2, 1,]

i = 0

執行到 first instruction: i = 0

一直行到 i = 0 那行就發生了錯誤，i 應該要為 len(x) - 1

count_positive(x):

辦不到，發生 error state 就一定導致 failure。

。

odd_or_pos(x):

辦不到，發生 error state 就一定導致 failure。

Fault:

* A static defect in software.
* Parts of source code that are incorrect.

Error State:

* An incorrect internal state.
* State information: variable values (including return value).

Failure:

* External, incorrect behavior with respect to the expected behavior.