Homework 1：Faulty Programs with Faults and Failures
Student：314581018 / 謝鎧駿

Part 1

以下是對 Homework 1 Part 1中四個程式的所有問題 (a)-(e) 的解答與分析，我會分別針對每個程式進行說明。

一、 Find last index of element

a. Describe the fault precisely by proposing a modification to the code.
   Fault：for (let i = x.length - 1; i > 0; i--) 應包含index 0，應改為
   i >= 0。
   Modification：for (let i = x.length - 1; i >= 0; i--) { if (x[i] === y) return i; }
b. give a test case that does not execute the fault.
   x = undefined; y = 2; → Expected = TypeError('The first parameter must be an array'), Actual = TypeError('The first parameter must be an array')
c. give a test case that executes the fault, but does not result in an error state.
   x = [2,3,5]; y = 5 → Expected = 2, Actual = 2
d. give a test case that results in an error state, but not a failure.
   impossible, 因為只要有error出現，就會連帶著出現failure，因為return值就會有錯
e. For the given test case in (d), describe the first error state. Be sure to describe the complete state.

二、 Find last index of zero

a. Describe the fault precisely by proposing a modification to the code.
   Fault：從頭找 0，但需找最後一個 0。
   Modification：for (let i = x.length - 1; i >= 0; i--) { if (x[i] === 0) return i; }
b. give a test case that does not execute the fault.
   x = undefined; Expected = TypeError('Not an array'), Actual = TypeError('Not an array')
c. give a test case that executes the fault, but does not result in an error state.

x = [1,0]; Expected = 1, Actual = 1

d. give a test case that results in an error state, but not a failure.

impossible, 因為只要有error出現，就會連帶著出現failure，因為return的index就會有錯

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

三、 Count positive elements

a. Describe the fault precisely by proposing a modification to the code.

Fault：條件 x[i] >= 0 包含 0。應改為 > 0。

Modification：if (x[i] > 0) count++;

b. give a test case that does not execute the fault.

x = undefined; → Expected = TypeError('Not an array');, Actual = TypeError('Not an array');

c. give a test case that executes the fault, but does not result in an error state.

impossible, 因為要有0出現，才會造成fault，但就會造成error，因為count變數有錯

d. give a test case that results in an error state, but not a failure.

impossible, 因為要有0出現，才會造成error，但就會造成failure，因為return的count就會與expected不同

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

四、 Count odd or postive elements

a. Describe the fault precisely by proposing a modification to the code.

Fault：x[i] % 2 === 1 對負數奇數會是 -1 而非 1。

Modification：if (Math.abs(x[i]) % 2 === 1 || x[i] > 0) 或 if (x[i] % 2 !== 0 || x[i] > 0)

b. give a test case that does not execute the fault.

x = undefined; → Expected = TypeError('Not an array'), Actual = TypeError('Not an array')

c. give a test case that executes the fault, but does not result in an error state.

Non, 因為要有負奇數出現，才會造成fault，但就會造成error，因為count變數就會有error

d. give a test case that results in an error state, but not a failure.

impossible, 因為要有負奇數出現，才會造成error，但就會造成failure，因為return的count就會不一樣

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

Part 2

一、 DataProcessor

每讀一行就new DataRecord並配置new char，但從未free memory，造成memory leak。因此開啟valgrind並編譯，讀取10000行後，valgrind會報告「definitely lost」的memory leak。

```
Starting file processing for: data.txt
Finished processing 10000 records.
==3579==
==3579== HEAP SUMMARY:
==3579==     in use at exit: 170,000 bytes in 20,000 blocks
==3579==   total heap usage: 20,007 allocs, 7 frees, 187,627 bytes allocated
==3579==
==3579== 170,000 (160,000 direct, 10,000 indirect) bytes in 10,000 blocks are definitely lost in loss record 2 of 2
==3579==    at 0x4C2B0E0: operator new(unsigned long) (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==3579==    by 0x401363: processLargeFile(std::string const&) (in /home/morris/Downloads/dataproc)
==3579==    by 0x401597: startDataIngestion() (in /home/morris/Downloads/dataproc)
==3579==    by 0x401632: main (in /home/morris/Downloads/dataproc)
==3579==
==3579== LEAK SUMMARY:
==3579==    definitely lost: 160,000 bytes in 10,000 blocks
==3579==    indirectly lost: 10,000 bytes in 10,000 blocks
==3579==      possibly lost: 0 bytes in 0 blocks
==3579==    still reachable: 0 bytes in 0 blocks
==3579==         suppressed: 0 bytes in 0 blocks
==3579==
==3579== For counts of detected and suppressed errors, rerun with: -v
==3579== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

Test Case :

直接開啟valgrind並編譯即可。

二、 LoggingSystem

多個 thread 同時對 total_logs_processed 進行遞增，未使用 mutex，導致race condition發生。

```
Running race condition example...
Expected total logs: 100000
Starting log processing with 10 threads.
Thread 7 starting to process logs...
Thread 8 starting to process logs...
Thread 0 starting to process logs...
Thread 9 starting to process logs...
Thread 6 starting to process logs...
Thread 5 starting to process logs...
Thread 4 starting to process logs...
Thread 3 starting to process logs...
Thread 2 starting to process logs...
Thread 1 starting to process logs...
Thread 9 finished.
Thread 8 finished.
Thread 1 finished.
Thread 6 finished.
Thread 2 finished.
Thread 3 finished.
Thread 7 finished.
Thread 4 finished.
Thread 0 finished.
Thread 5 finished.

All threads have finished.
Final count of logs processed: 99993
```

Test Case ：

直接編譯並執行即可發現Final count of logs processed 竟然會小於100000，表示有race condition發生。

三、 MatrixProcessor

const_cast<int**> 之後，將指標直接assign 999，這是無效指標，造成未定義行為/Segmentation Fault。

```
MatrixProcessor.cpp: In function ‘void process_matrix(const int* const*, int, int)’:
MatrixProcessor.cpp:27:22: error: invalid conversion from ‘int’ to ‘int**’ [-fpermissive]
     non_const_matrix = 999;
                        ^
```

Test Case ：

直接編譯就會報錯了。

四、 ProfileUpdater

UserProfile.username 陣列大小只有1，但 strcpy 可以複製任意長度字串，產生Buffer Overflow。所以當我們用AddressSanitizer檢測時會報告Error。

```
Authenticating user...
Updating user profile with new username...
=================================================================
==3673== ERROR: AddressSanitizer: unknown-crash on address 0x7fff130123c0 at pc 0x7fc2cdee4755 b
p 0x7fff13012350 sp 0x7fff13011b10
```

Test Case ：

在main中呼叫時傳入一個很長的字串並開啟AddressSanitizer

```
// The main function demonstrating the vulnerability.
int main() {
    // trigger the fault
    UserProfile p;
    updateUserProfile(p, std::string(100, 'A'));
    printProfile(p);
    return 0;
}
```

五、 ResourceScheduler

當兩個 thread 鎖的順序不同：一個先鎖 A 再鎖 B，另一個先鎖 B 再鎖 A，會產生Deadlock，如下圖所示。

```
2025-09-21 16:57:42,370 | Thread-1 (worker_thread_a) | Thread-A is starting.
2025-09-21 16:57:42,370 | Thread-1 (worker_thread_a) | Thread-A attempting to acquire lock on Resource A...
2025-09-21 16:57:42,370 | Thread-2 (worker_thread_b) | Thread-B is starting.
2025-09-21 16:57:42,372 | Thread-1 (worker_thread_a) | Thread-A acquired lock on Resource A. Waiting for Re
source B...
2025-09-21 16:57:42,372 | Thread-2 (worker_thread_b) | Thread-B attempting to acquire lock on Resource B...
2025-09-21 16:57:42,372 | Thread-2 (worker_thread_b) | Thread-B acquired lock on Resource B. Waiting for Re
source A...
```

Test Case :

只需要在main中分別宣告兩個worker thread並start他們就會發現fault了，而task queue不管放甚麼都不會影響到此Fault。

```python
# The main function to set up and run the threads.
if __name__ == "__main__":
    t1 = threading.Thread(target=worker_thread_a, args=("Thread-A",))
    t2 = threading.Thread(target=worker_thread_b, args=("Thread-B",))
    t1.start()
    t2.start()
    t1.join()
    t2.join()
```