Report3

Chap 14 (Debugging Memory Allocation w/ valgrind)

Question 1.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(){
5       int* x;
6       x = NULL;
7       printf("%d", *x);
8 }
```

Line 5 : creates a pointer to an integer

Line 6 : sets it to NULL

Line 7 : tries to dereference it

```
OperatingSystems [main] ⚡ ./a.out
[1]    3921 segmentation fault (core dumped)  ./a.out
```

Segmentation fault 가 일어났다.

Question 2, 3.

-g flag 를 포함해 compile 한 후 valgrind 를 사용한 결과이다.

```
OperatingSystems [main] ⚡ valgrind --leak-check=yes ./a.out
==4870== Memcheck, a memory error detector
==4870== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==4870== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==4870== Command: ./a.out
==4870==
==4870== Invalid read of size 4
==4870==    at 0x109161: main (null.c:7)
==4870==  Address 0x0 is not stack'd, malloc'd or (recently) free'd
==4870==
==4870==
==4870== Process terminating with default action of signal 11 (SIGSEGV)
==4870==  Access not within mapped region at address 0x0
==4870==    at 0x109161: main (null.c:7)
==4870==  If you believe this happened as a result of a stack
==4870==  overflow in your program's main thread (unlikely but
==4870==  possible), you can try to increase the size of the
==4870==  main thread stack using the --main-stacksize= flag.
==4870==  The main thread stack size used in this run was 8388608.
==4870==
==4870== HEAP SUMMARY:
==4870==     in use at exit: 0 bytes in 0 blocks
==4870==   total heap usage: 0 allocs, 0 frees, 0 bytes allocated
==4870==
==4870== All heap blocks were freed -- no leaks are possible
==4870==
==4870== For lists of detected and suppressed errors, rerun with: -s
==4870== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
[1]    4870 segmentation fault (core dumped)  valgrind --leak-check=yes ./a.out
```

Invalid read of size 4 라는 결과가 나온 것을 알 수 있다.

7 번 줄에서 dereference 를 시도했을 때, x 라는 포인터가 NULL 이므로 x 포인터가 가리키는 값을 프린트하려 했을 때, 유효하지 않은 읽음이 발생한 것이다.

Malloc()은 한번도 사용하지 않았으므로 하단에 HEAP SUMMARY 에는 아무런 문제도 없는 것을 볼 수 있다.

Question 4.

```c
#include <stdio.h>
#include <stdlib.h>

int main(){
    int* x = (int*)malloc(sizeof(int));
}
```

Line 5 : allocates memory using malloc()

하지만 그 exit 하기 전 free()해주지 않았다.

프로그램은 문제없이 compile 되었고 run 되었다.

Valgrind 를 사용해 확인해 보았다.

```
 OperatingSystems [main]  valgrind --leak-check=yes ./a.out
==23458== Memcheck, a memory error detector
==23458== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==23458== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==23458== Command: ./a.out
==23458==
==23458==
==23458== HEAP SUMMARY:
==23458==     in use at exit: 4 bytes in 1 blocks
==23458==   total heap usage: 1 allocs, 0 frees, 4 bytes allocated
==23458==
==23458== 4 bytes in 1 blocks are definitely lost in loss record 1 of 1
==23458==    at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgprelo
ad_memcheck-amd64-linux.so)
==23458==    by 0x10915E: main (in /home/soohun/Desktop/coconut/OperatingSystems
/a.out)
==23458==
==23458== LEAK SUMMARY:
==23458==    definitely lost: 4 bytes in 1 blocks
==23458==    indirectly lost: 0 bytes in 0 blocks
==23458==      possibly lost: 0 bytes in 0 blocks
==23458==    still reachable: 0 bytes in 0 blocks
==23458==         suppressed: 0 bytes in 0 blocks
==23458==
==23458== For lists of detected and suppressed errors, rerun with: -s
==23458== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

HEAP SUMMARY 를 보면 4 byte 가 in use at exit 인 것을 알 수 있다.

Sizeof(int), 즉 4byte 를 malloc()을 통해 할당했지만 free 하지 않은 결과이다.

Total heap usage 를 보면 1 allocs, 0 frees, 4 bytes allocated 이다.

마지막 LEAK SUMMARY 를 보면 definitely lost, 즉 결과적으로 일어난 memory leak 은 4byte 임을 알 수 있다.

Question 5.

```c
#include <stdio.h>
#include <stdlib.h>

int main(){
    int* data = (int*)malloc(100);
    data[100] =0;
}
```

Line 5 : creates an array of integers called data of size 100 using malloc

Line 6 : sets data[100] to zero

위 코드를 compile 후 run 해보았을 때, 문제없이 run 되었다. Valgrind 로 확인한 결과는 아래와 같다.

```
☁ OperatingSystems [main] ⚡ valgrind --leak-check=yes ./a.out
==23762== Memcheck, a memory error detector
==23762== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==23762== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==23762== Command: ./a.out
==23762==
==23762== Invalid write of size 4
==23762==    at 0x10916D: main (in /home/soohun/Desktop/coconut/OperatingSystems
/a.out)
==23762==  Address 0x4a591d0 is 224 bytes inside an unallocated block of size 4,
194,032 in arena "client"
==23762==
==23762==
==23762== HEAP SUMMARY:
==23762==     in use at exit: 100 bytes in 1 blocks
==23762==   total heap usage: 1 allocs, 0 frees, 100 bytes allocated
==23762==
==23762== 100 bytes in 1 blocks are definitely lost in loss record 1 of 1
==23762==    at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgprelo
ad_memcheck-amd64-linux.so)
==23762==    by 0x10915E: main (in /home/soohun/Desktop/coconut/OperatingSystems
/a.out)
==23762==
==23762== LEAK SUMMARY:
==23762==    definitely lost: 100 bytes in 1 blocks
==23762==    indirectly lost: 0 bytes in 0 blocks
==23762==      possibly lost: 0 bytes in 0 blocks
==23762==    still reachable: 0 bytes in 0 blocks
==23762==         suppressed: 0 bytes in 0 blocks
==23762==
==23762== For lists of detected and suppressed errors, rerun with: -s
==23762== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 0 from 0)
```

Invalid write of size 4 가 발생한 것을 알 수 있다.

그 밑에 보면 size 4 의 unallocated block 이 있다는 것을 볼 수 있었다. data 라는 배열은 malloc(100)을 통해 heap 공간에 allocate 된 100 byte 즉, integer 25 개의 data[24]까지만 allocate 된 메모리이다. 하지만 data[100]이라는 할당되지 않은 공간에 데이터를 write 하려 했으므로 이런 오류 메시지가 표시된 것이다.

아래의 HEAP SUMMARY 를 보면 100byte 가 in use at exit 이다. Malloc 을 통해 100byte 를 할당했지만 후에 free 를 안 해주었으므로 당연한 결과이다.

LEAK SUMMARY 에서 100byte 의 memory leak 이 발생한 것을 확인할 수 있었다.

Question 6.

```c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(){
5     int* array = (int*)malloc(100*sizeof(int));
6     free(array);
7     printf("%d", array[50]);
8 }
```

Line 5 : allocates an array of integers

Line 6 : frees them

Line 7 : print the value of one of the elements of the array

Compile 과 run 은 실행되었다. Valgrind 결과는 아래와 같다.

```
☁ OperatingSystems [main] ⚡  valgrind --leak-check=yes ./a.out
==24441== Memcheck, a memory error detector
==24441== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==24441== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==24441== Command: ./a.out
==24441==
==24441== Invalid read of size 4
==24441==    at 0x1091B9: main (in /home/soohun/Desktop/coconut/OperatingSystems
/a.out)
==24441==  Address 0x4a59108 is 200 bytes inside a block of size 400 free'd
==24441==    at 0x483CA3F: free (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload
_memcheck-amd64-linux.so)
==24441==    by 0x1091AE: main (in /home/soohun/Desktop/coconut/OperatingSystems
/a.out)
==24441==  Block was alloc'd at
==24441==    at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgprelo
ad_memcheck-amd64-linux.so)
==24441==    by 0x10919E: main (in /home/soohun/Desktop/coconut/OperatingSystems
/a.out)
==24441==
0==24441==
==24441== HEAP SUMMARY:
==24441==     in use at exit: 0 bytes in 0 blocks
==24441==   total heap usage: 2 allocs, 2 frees, 1,424 bytes allocated
==24441==
==24441== All heap blocks were freed -- no leaks are possible
==24441==
==24441== For lists of detected and suppressed errors, rerun with: -s
==24441== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

Invalid read of size4 가 발생했다.

Free 한 heap memory 는 접근할 수 없다. 하지만 그 array 의 부분을 print 하려 했으므로, 즉 read 하려 했으므로 이런 에러 메시지가 표시된 것이다.

하지만 HEAP SUMMARY 를 보면 All heap blocks were freed – no leaks are possible 인 것을 볼 수 있다.

즉 memory 접근에서는 문제가 있지만 free 를 해주었기 때문에 memory leak 은 발생하지 않았다. 코드 상에서의 malloc()은 하나지만 total heap usage 에서 2 allocs 라고 나오는 이유는, printf 와 같이 stdout 으로 출력할 때, mmap 이 아닌 malloc 을 사용하도록 되어있기 때문이다. 즉 printf() 부분에서 내부적으로 한번의 alloc 과 free 가 이뤄진 것이라고 볼 수 있다.

Question 7.

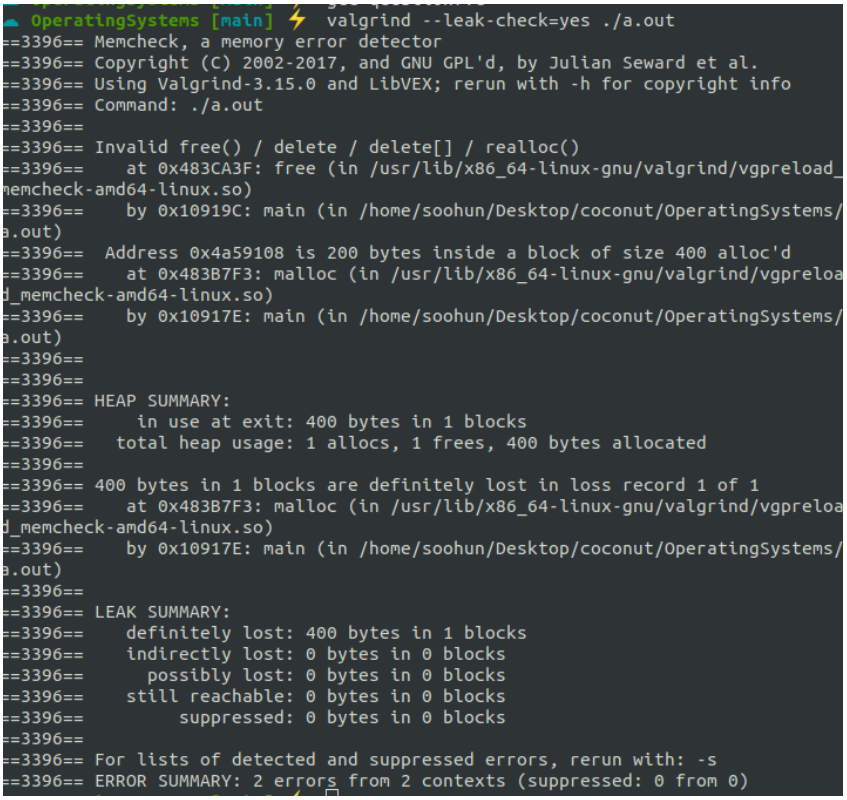
```c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(){
5     int* array = (int*)malloc(100*sizeof(int));
6     int* ptr;
7     ptr = &array[50];
8     free(ptr);
9
0 }
```

Line 5 : allocates an array of 100 * 4bytes

Line 6 : creates a pointer ptr

Line 7 : ptr gets an address of 50th index of array

Line 8 : free 50th index of array / pass a funny value to free



invalid free 라는 에러 메시지를 볼 수 있다.

Heap memory 에 할당된 공간 중 중간에 일부분만 free 하는 것은 불가능하다.

그래서 밑에 자세히 보면 200 bytes inside a block of size 400 alloc'd 인 라는 문구를 볼 수 있다.

HEAP SUMMARY 를 보면 횟수로는 1 allocs, 1frees 라고 나와 문제가 없어 보이지만, free()가 invalid 했으므로 밑의 LEAK SUMMARY 를 보면 400 bytes 의 memory leak 이 일어난 것을 확인할 수 있다.