[OS'25] Project Testing Cases

A- Instructions

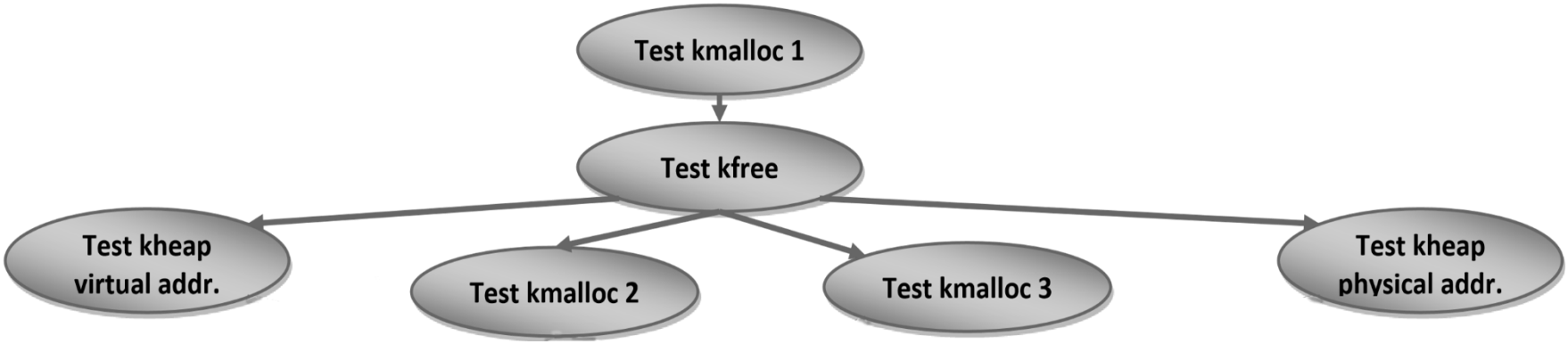
1. Test each part from the project independently.
2. After completing all parts, test the whole project.
3. The individual tests MUST meet the following time limits:

    1. *tstkvirtaddr (k virtual address test):* ***max of 3 min / each***
    2. *All other individual tests****: max of 1 min / each***

4. During your solution, don't change any file EXCEPT those who contain "TODO",
5. In bonuses & challenges, if you change any other file during your solution, kindly MAKE SURE to tell us when you deliver the code.

B- Dependency Graph of Ready-Made Tests

The following graph shows the dependencies between the ready-made tests.



C- Responsibility of Each Ready-Made Test

The following tables show the main points that each of the test programs will check for!!

Kmalloc (1 & 2)	Kfree	Kheap_virtual_address	Kheap_physical_address
1. Return addr. (4KB boundary) 2. Memory allocation (Strategy) 3. Page File allocation (nothing) 4. memory access (R & W) 5. Insufficient space 6. Permissions	1. Memory de-allocation 2. Tables of KHEAP (exists) 3. Memory access after free 4. Del. Non-exist variable 5. Allocation after free	1. Get va after kmalloc only 2. Get va after kmalloc & kfree 3. Get va of frames that are not belong to KHEAP	1. Get pa after kmalloc only 2. Get pa after kmalloc & kfree 3. Get pa of non-exist area

D- Testing Procedures

FIRST: Testing Each Part

Run every test of the following. If a test succeeds, it will print and success message on the screen, otherwise the test will panic at the error line and display it on the screen.

> **IMPROTANT NOTES:**
>
> 1. Run each test in **NEW SEPARATE RUN**
> 2. If the test of certain part is failed, then there's a problem in your code
> 3. Else, this NOT ensures 100% that this part is totally correct. So, make sure that your logic matches the specified steps exactly

1. Testing KERNEL Heap:

tstkmalloc1 command: tests the implementation of **kmalloc()**. It validates return addresses from the kmalloc(),number of allocated frames, accessing the allocated space and permissions (First Fit)

- **FOS>** `tstkmalloc 1`

tstkmalloc2 command: tests the implementation of **kmalloc()** (kfree must be implemented in order to run this test). It validates return addresses from the kmalloc(), testing the First fit strategy by creating some holes in the memory using **kfree().**

YOU WRITE ONE OF THE FOLLOWING 4 LINES BASED ON YOUR STARTEGY

- **FOS>** `khfirstfit //if your strategy is first fit`
- **FOS>** `khbestfit //if your strategy is best fit`
- **FOS>** `khnextfit //if your strategy is next fit`
- **FOS>** `khworstfit //if your strategy is worst fit`
- **FOS>** `tstkmalloc 2`

tstkfree command: tests the implementation of **kfree()**. It validates the number of freed frames by kfree(). It checks the memory access (read & write) of the removed spaces and allocation after free. Also, it ensure that KHEAP tables are not removed.

YOU WRITE ONE OF THE FOLLOWING 4 LINES BASED ON YOUR STARTEGY

- **FOS>** `khfirstfit //if your strategy is first fit`
- **FOS>** `khbestfit //if your strategy is best fit`
- **FOS>** `khnextfit //if your strategy is next fit`
- **FOS>** `khworstfit //if your strategy is worst fit`
- **FOS>** `tstkfree`

tstkvirtaddr command: tests the implementation of **kheap_virtual_address()**. It validates the returned virtual address of the given physical one for three cases: 1. After kmalloc only, 2. After kmalloc and kfree, 3. For frames that are not belong to KERNEL HEAP (should return 0).

- **FOS>** `tstkvirtaddr`

tstkphysaddr command: tests the implementation of **kheap_physical_address()**. It validates the returned physical address of the given virtual one for three cases: 1. after kmalloc only, 2. after kmalloc and kfree, 3. for not allocated area in KERNEL HEAP (should return 0).

- **FOS>** `tstkphysaddr`

Enjoy writing your own OS

GOOD LUCK