Operating Systems’25

Project Description

Agenda

- Logistics
- What’s New?
- [**Kernel**] Project Features
    1. Kernel Heap
- Project Quick Guide

Logistics

- **Startup Code**:
  - FOS_PROJECT_2025_template.zip
  - Follow these steps to import the project folder into the eclipse
  - The **ONLY** functions that should be implemented contains the following comment:

```
//TODO: [PROJECT 2025] …
```

ADVICES

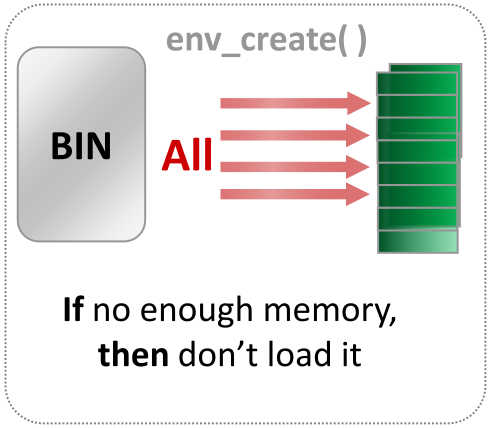
- **#1: Work as a Team**
  - Task division guide may help
- **#2: Start Immediately**
  - To get benefit of the support
- **#3: Read docs & ppt**
  - Detailed steps & helper functions
- **#4: Read and Adhere to Instructions**
  - To successfully deliver your project

Delivery

- **Submission: DROPBOX BASED**
- **Test cases** will be used to evaluate your solution
- Each case **is binary**: success (1) or not (0)
- **Make sure** they are run correctly before you deliver isA
- **Delivery Dates:**
    - **Final Delivery: THU (*17 APRIL @10:00 PM*)**
- **ONE MILESTONE** IS **FINAL** delivery
    - **MUST** deliver the required tasks and **ENSURE** they're worked correctly
- **Support:** *WEEKLY OFFICE HOURS*

What's New?

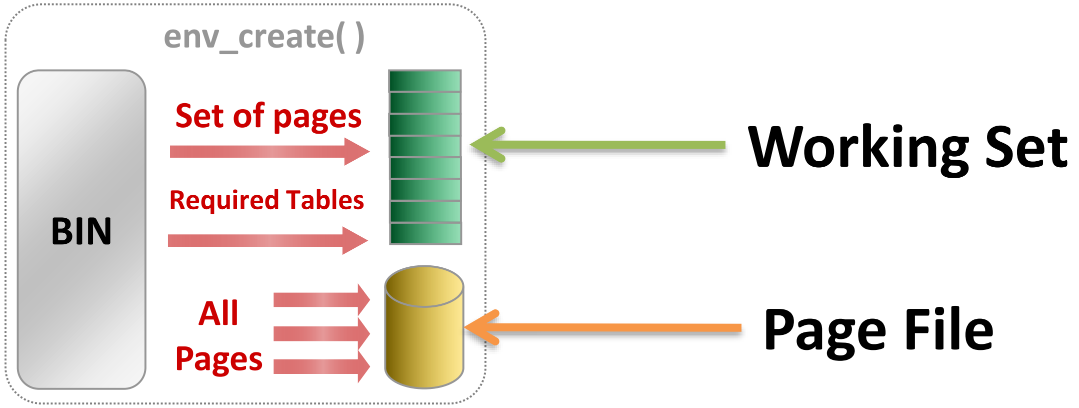
OLD



If no enough memory, **then** don't load it

NEW

NEW Concepts



Refer to the

Project Documentation

MS1 Features

[KERNEL]

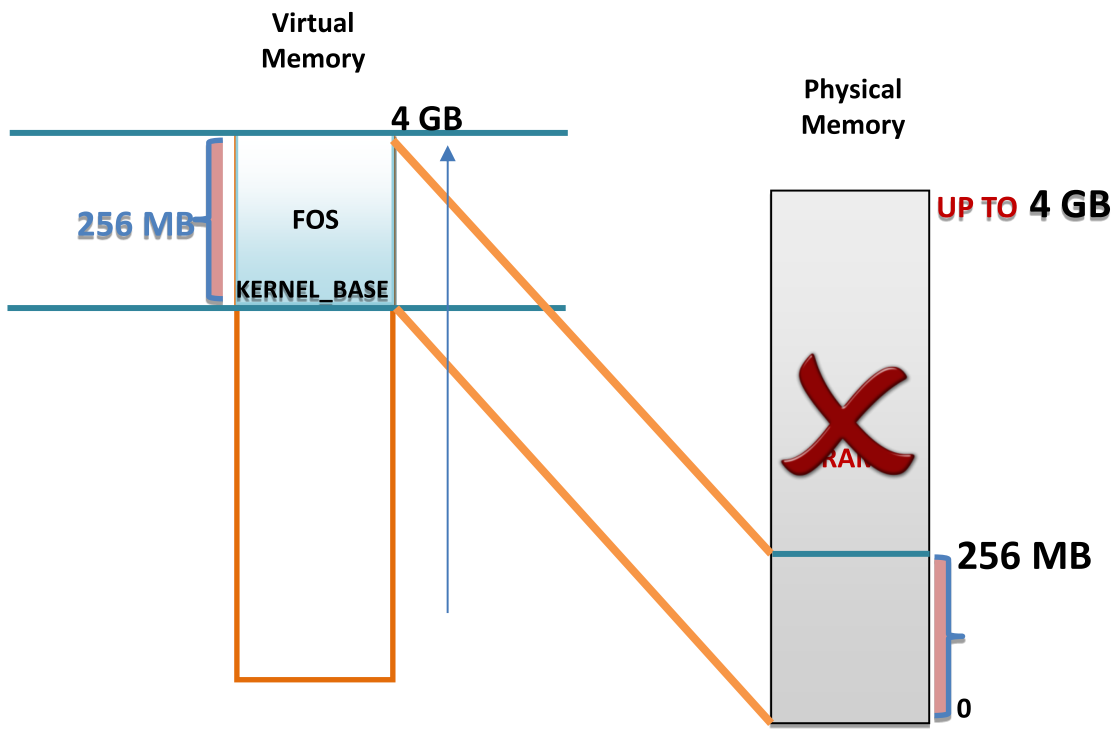
1. **Kernel Heap:** dynamic allocation and free

   Using one of the following strategies: (NEXT FIT, FIRST FIT, WORST FIT, BEST FIT)

   The required strategy of each team will be sent to each team to the registered email.

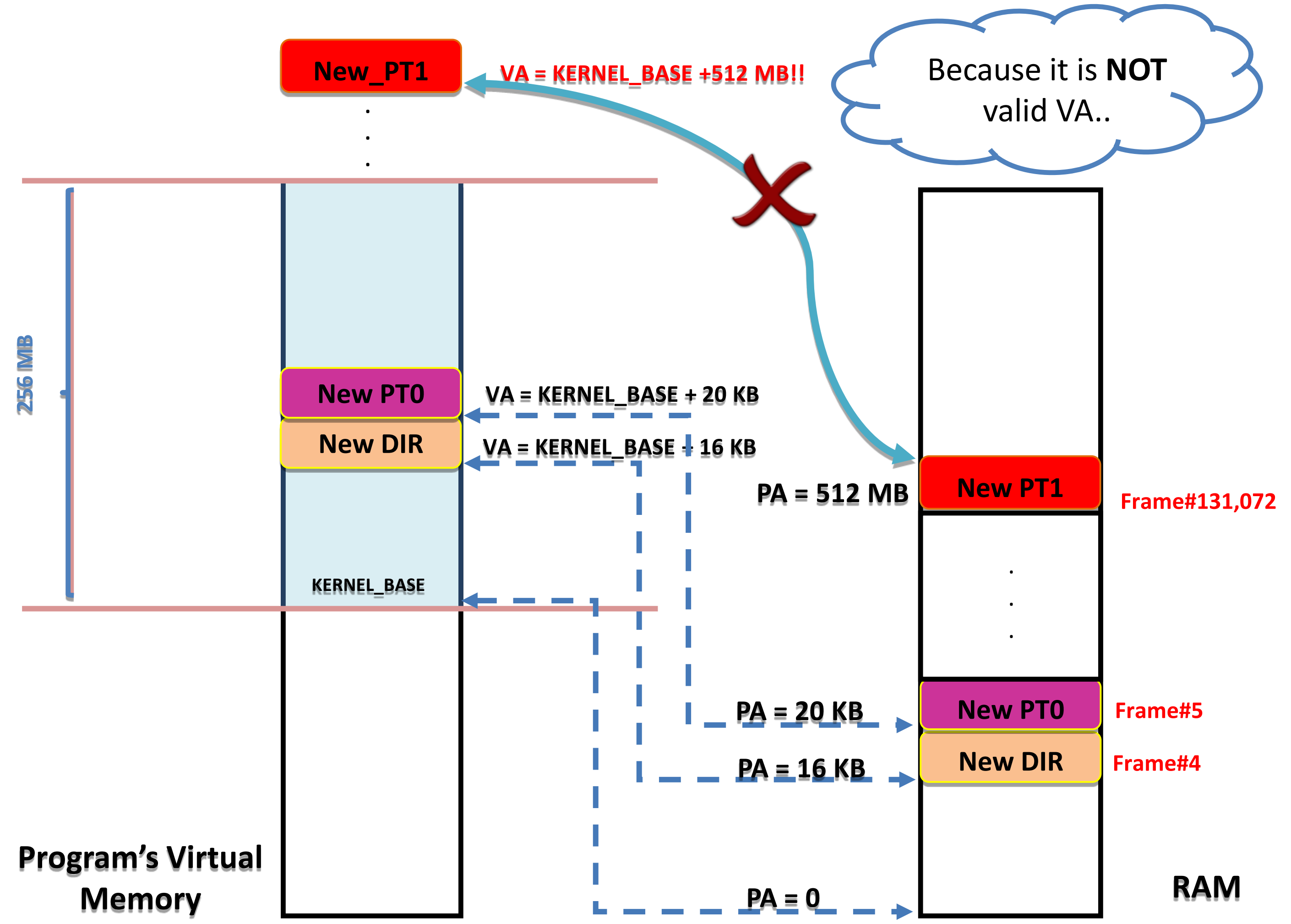
Kernel Heap

- Current: Kernel is **one-to-one** mapped to 256 MB RAM
- Problem: Kernel can't directly access beyond 256 MB RAM



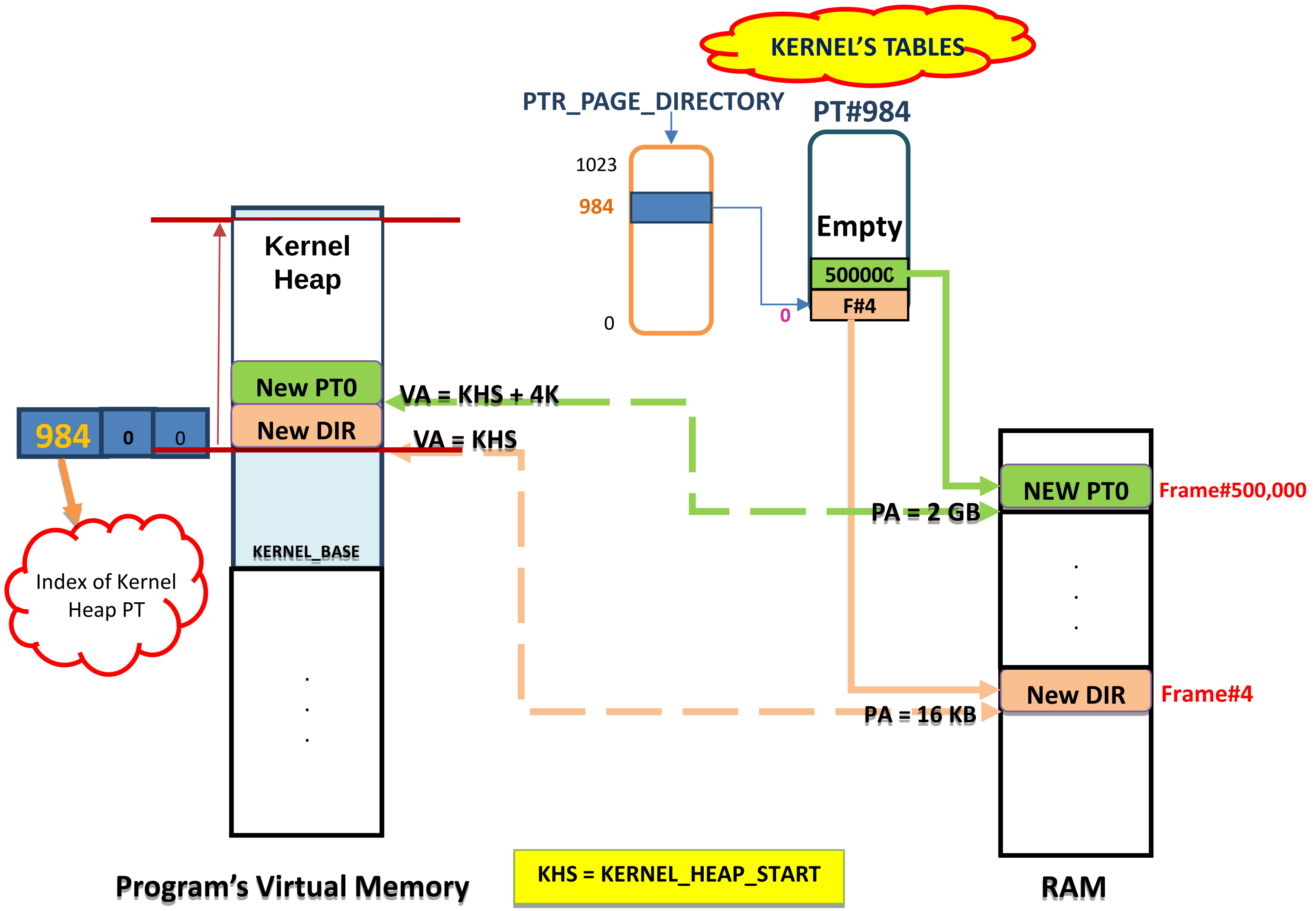
Kernel Heap

- Example: Kernel can't directly access beyond 256 MB RAM



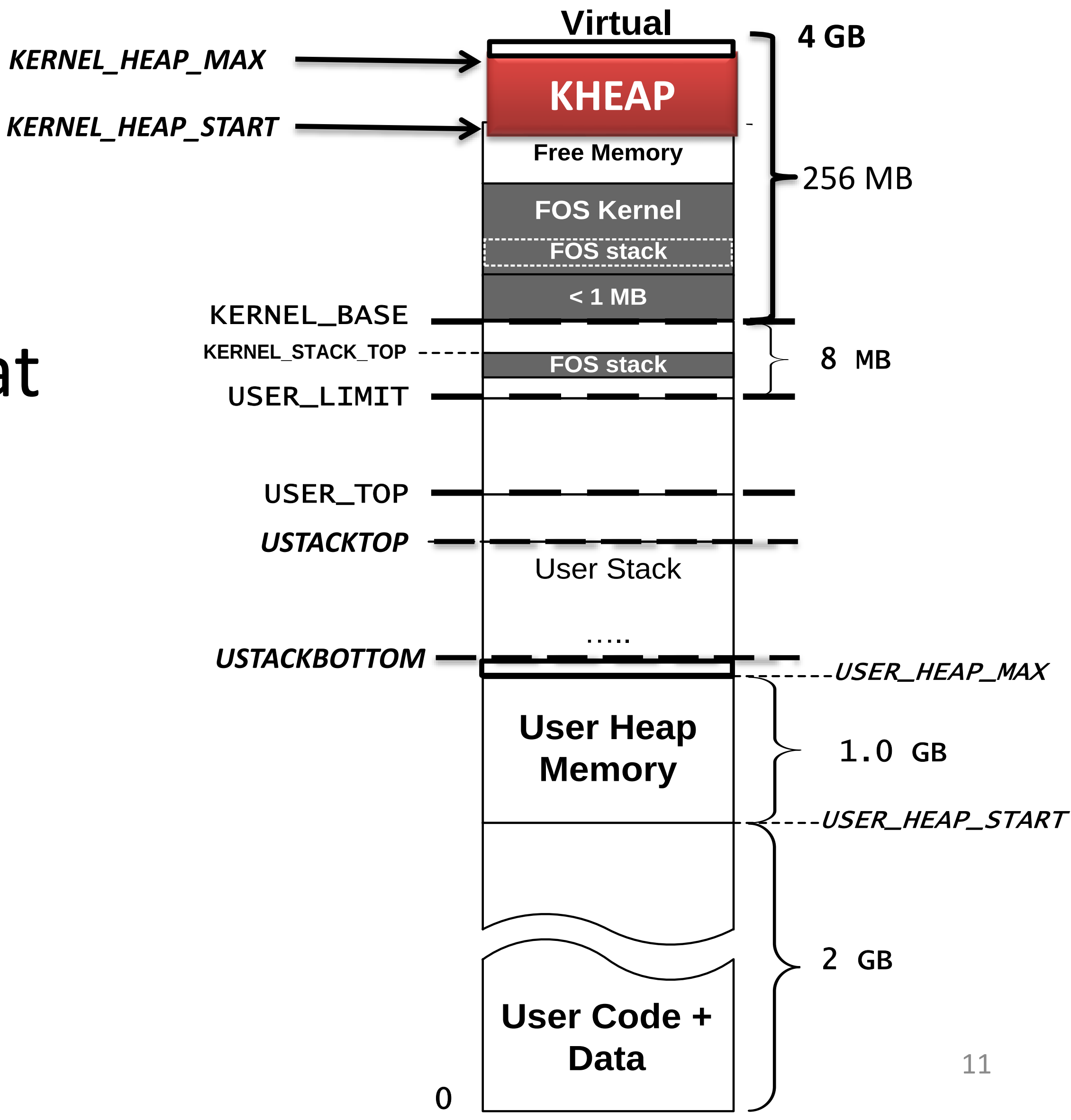
Kernel Heap

- **Solution:** Kernel Heap for dynamic allocations (**No 1-1 map**)



Kernel Heap

- Kernel Heap lies at the end of the virtual space



Kernel Heap

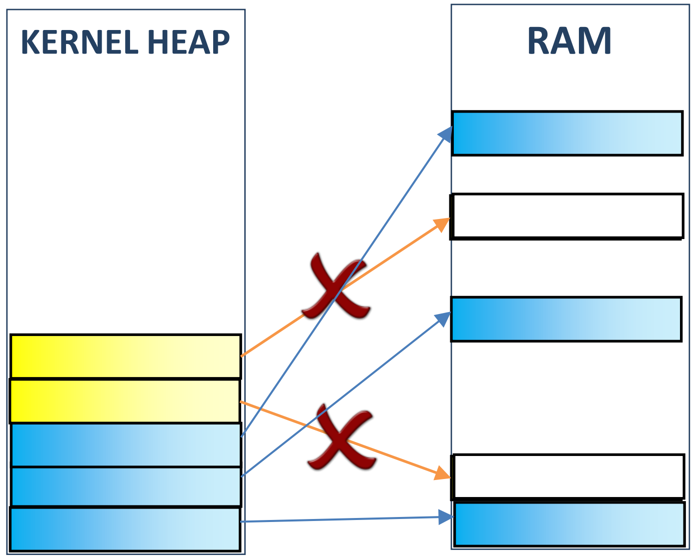
1. **Kmalloc():** dynamically allocate space
2. **Kfree():** delete a previously allocated space

kmalloc()

Kfree()

ALL Pages Created in Memory

Remove Pages BUT NOT Tables



Kernel Heap

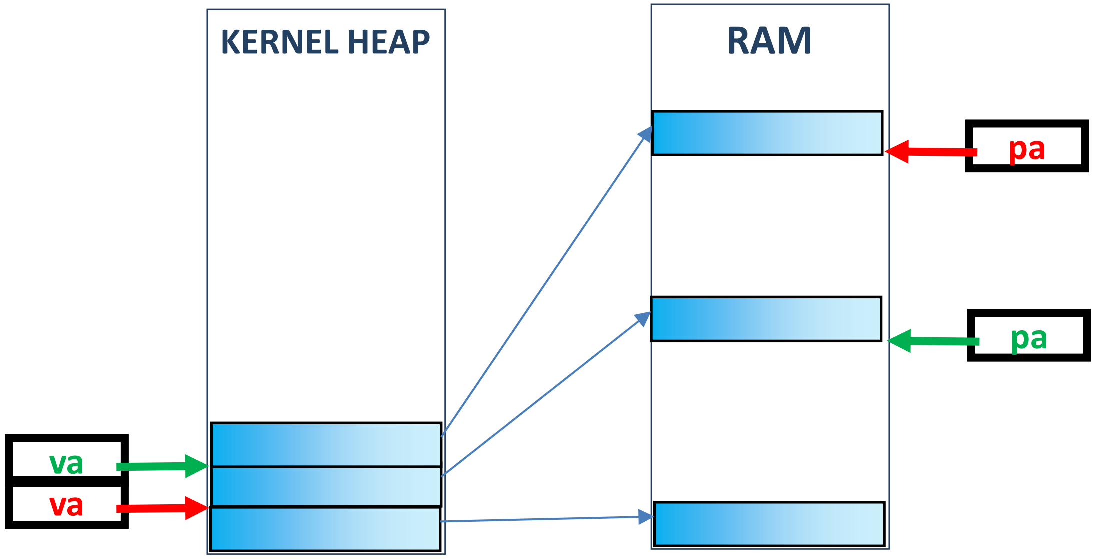
3. **Kheap_physical_address():** find physical address of the given kernel virtual address
4. **Kheap_virtual_address():** find kernel virtual address of the given physical one

kheap_physical_address()

kheap_virtual_address()

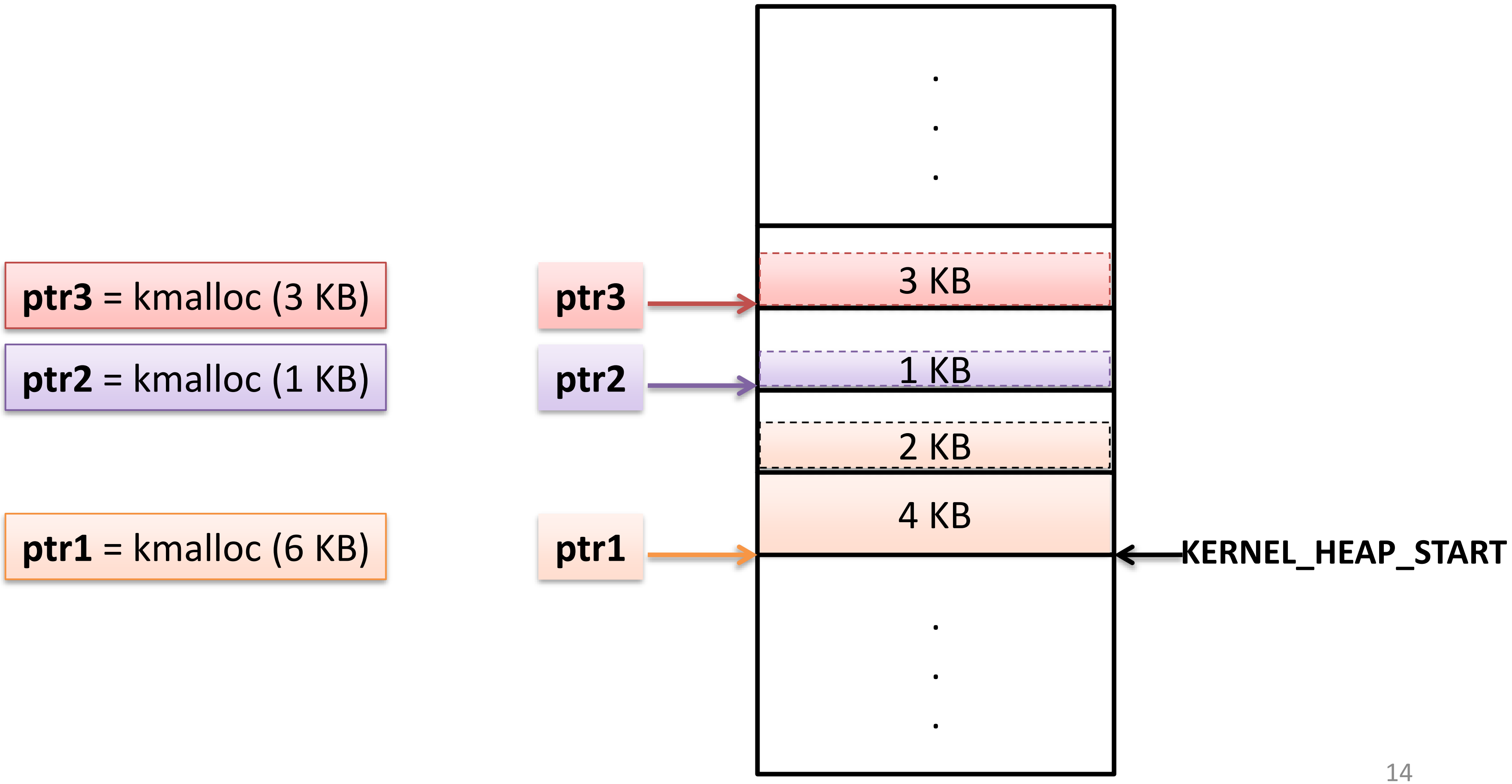
Get pa of the given va

Get va of the given pa



Kernel Heap
[kmalloc() / kfree()]

- **Allocate pages on 4KB granularity**



Dynamic allocation/Deallocation [kmalloc() / kfree()]

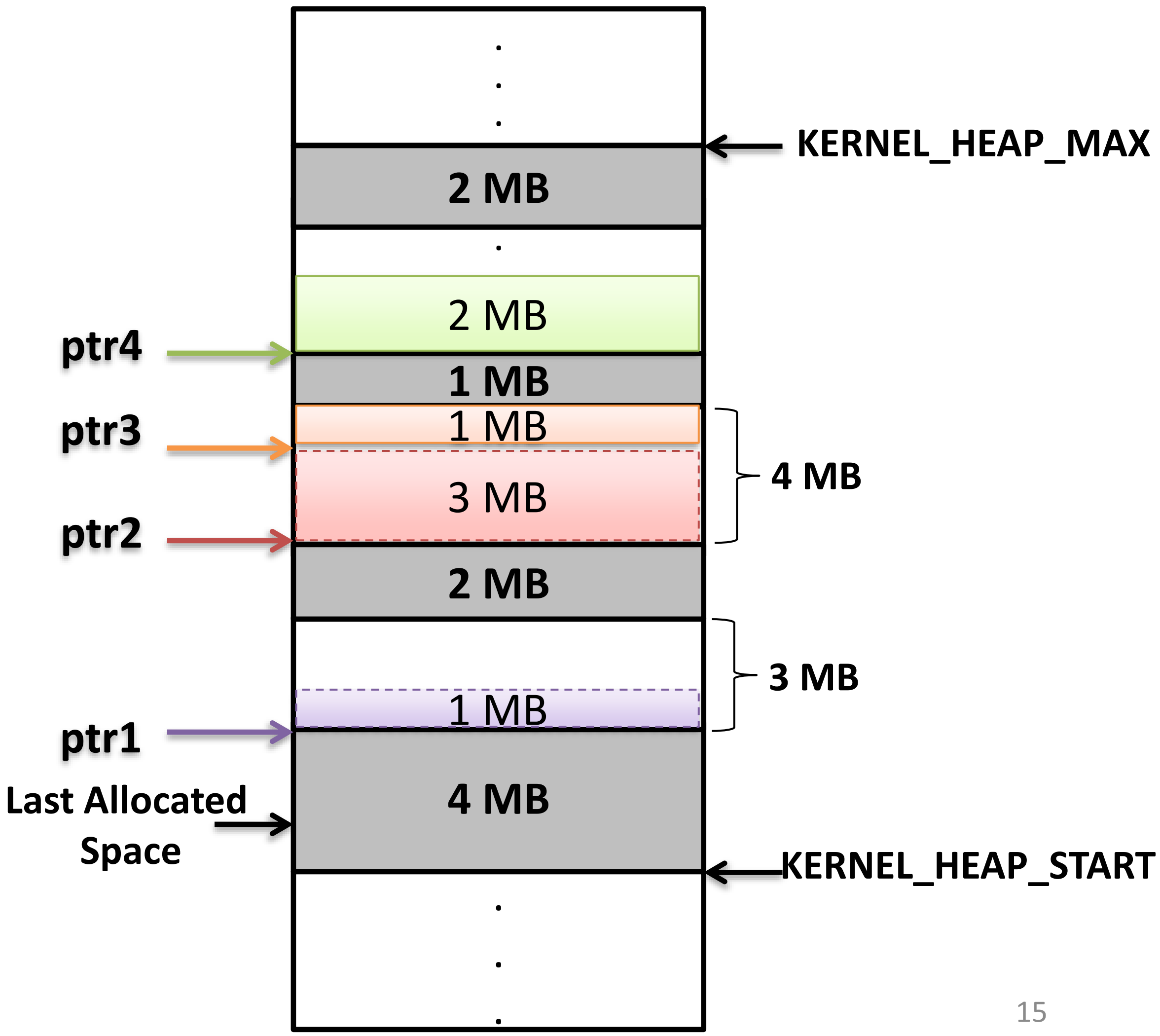
NEXT FIT Strategy

ptr4 = kmalloc (2 MB)

ptr3 = kmalloc (1 MB)

ptr2 = kmalloc (3 MB)

ptr1 = kmalloc (1 MB)



Dynamic allocation/Deallocation [kmalloc() / kfree()]

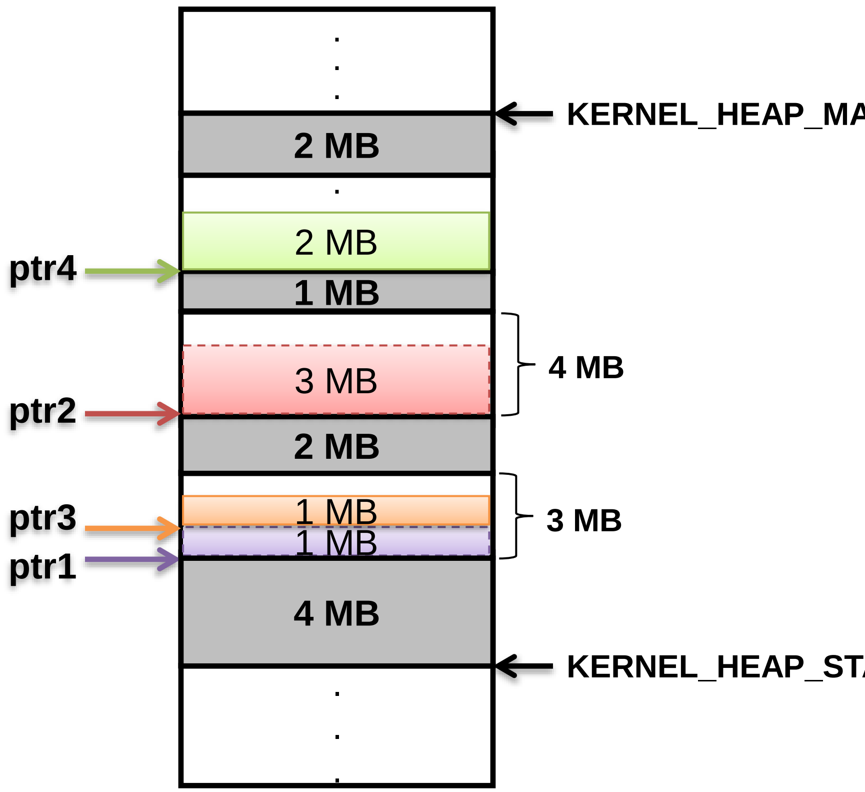
FIRST FIT Strategy

ptr4 = kmalloc (2 MB)

ptr2 = kmalloc (3 MB)

ptr3 = kmalloc(1 MB)

ptr1 = kmalloc (1 MB)



Dynamic allocation/Deallocation [kmalloc() / kfree()]

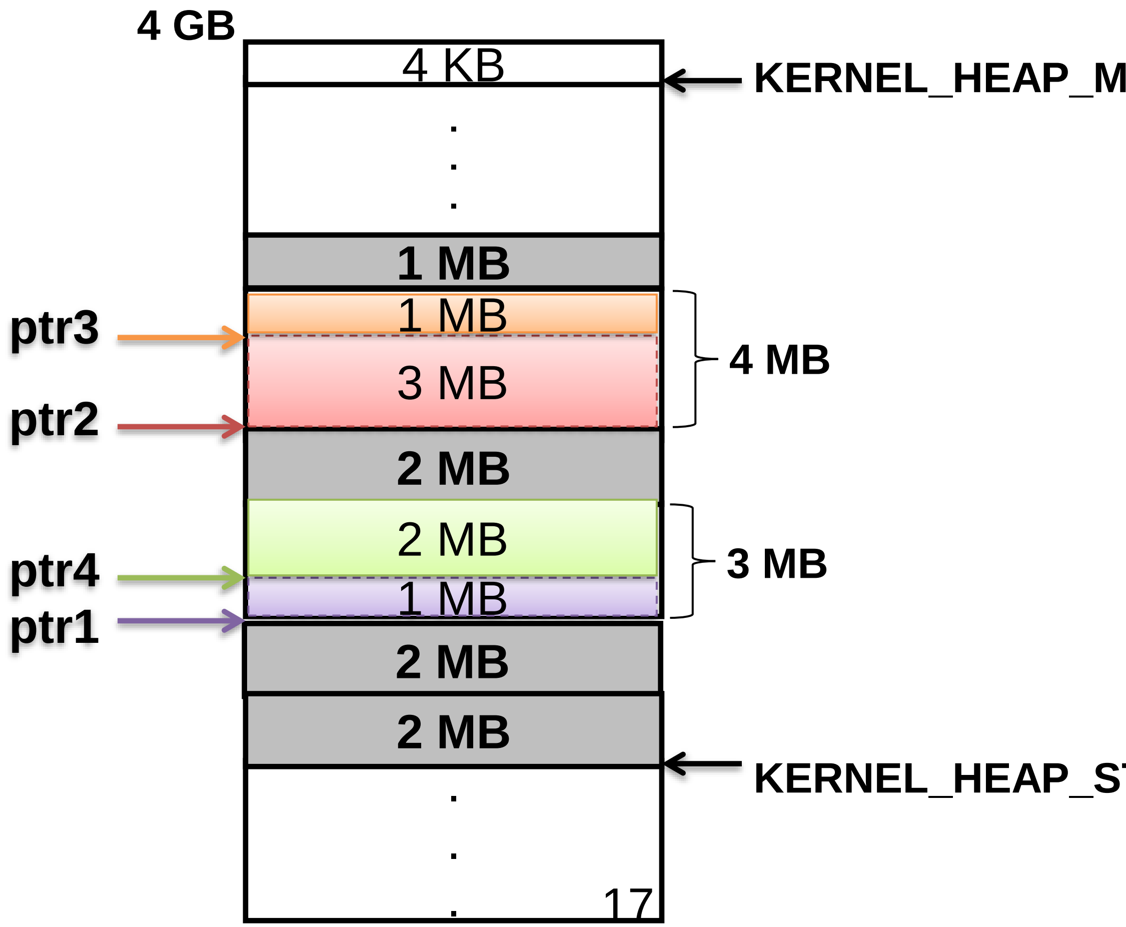
BEST FIT Strategy

ptr3 = kmalloc (1 MB)

ptr2 = kmalloc (3 MB)

ptr4 = kmalloc (2 MB)

ptr1 = kmalloc (1 MB)



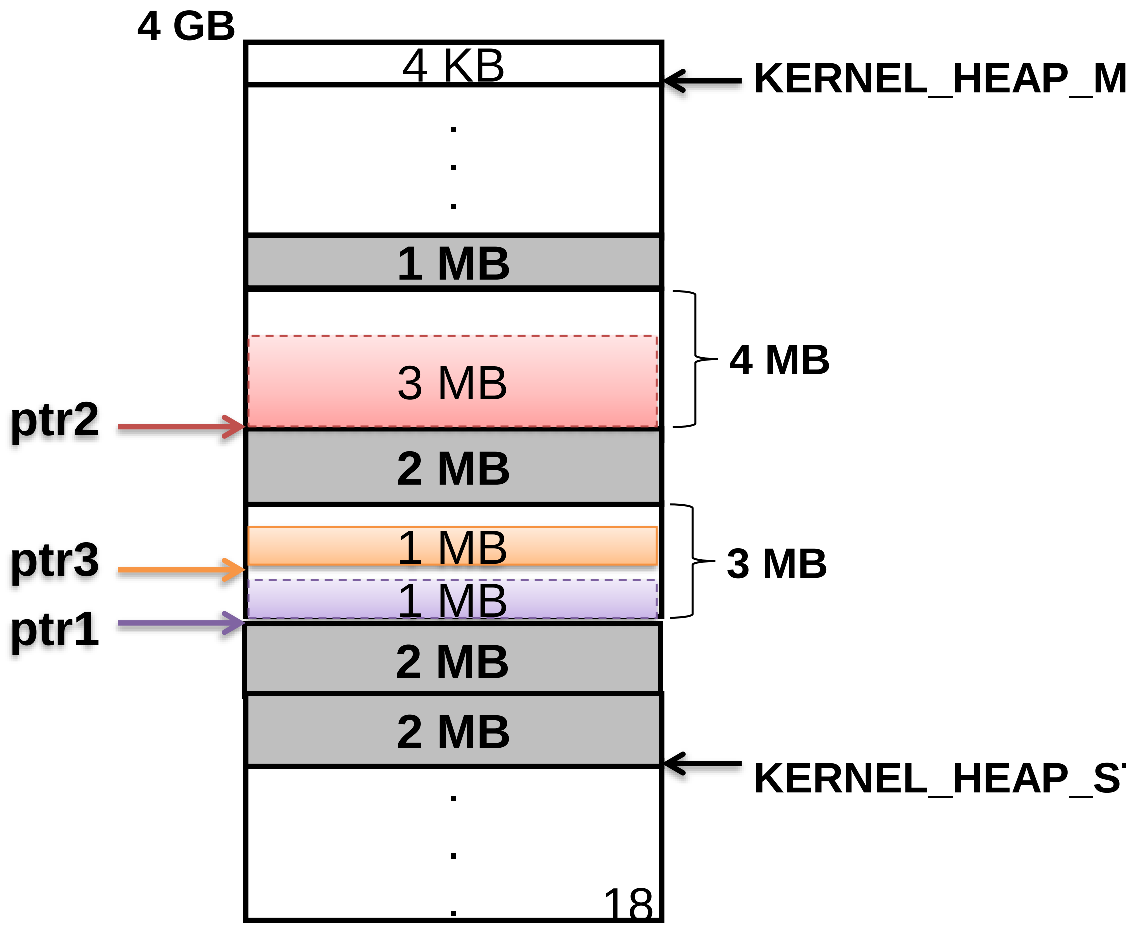
Dynamic allocation/Deallocation [kmalloc() / kfree()]

WORST FIT Strategy

ptr2 = kmalloc (3 MB)

ptr3 = kmalloc (1 MB)

ptr1 = kmalloc (1 MB)



Startup Code

FOS_PROJECT_2025_Template.Zip

Follow these steps to import the project folder into the eclipse

ALL Required Functions

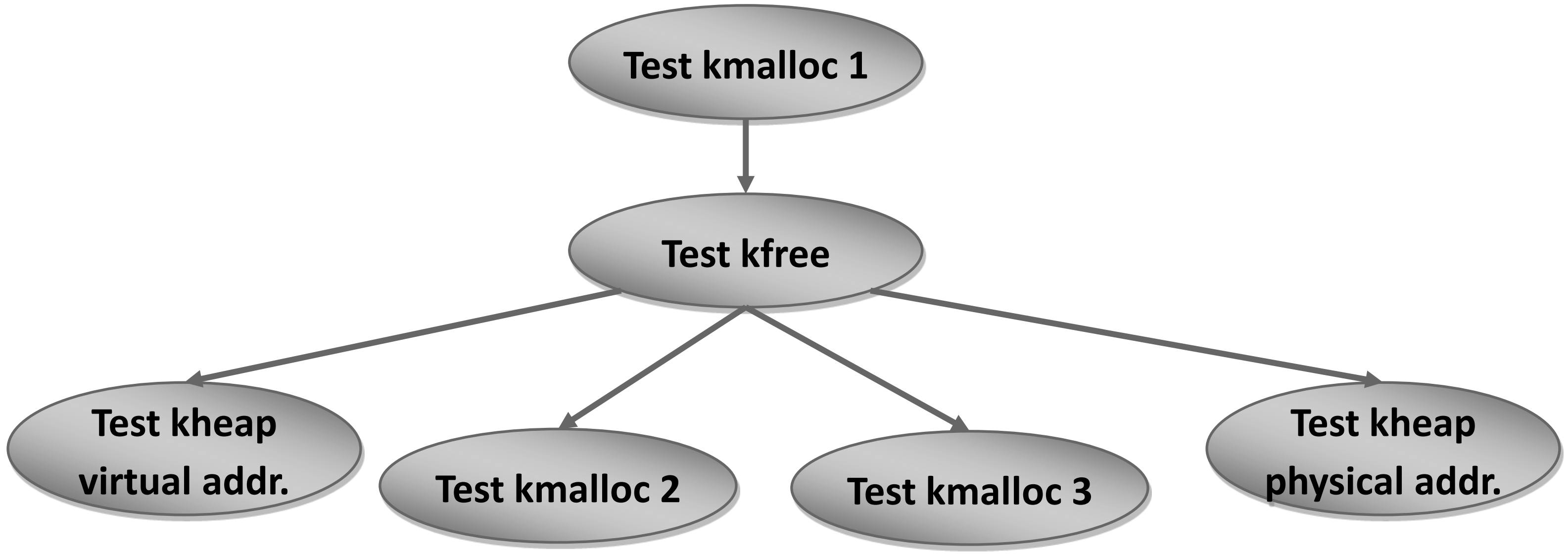
1. Kernel Heap

MAIN Functions	
`Kmalloc`	**Test 1: FOS>** `tstkmalloc 1` **Test 2: FOS>** `tstkmalloc 2` //Depend on kfree **Test 3: FOS>** `tstkmalloc 3` //Depend on kfree
`Kfree`	**Test 1: FOS>** `tstkfree`
`kheap_virtual_address`	**Test 1: FOS>** `tstkvirtaddr`
`kheap_physical_address`	**Test 1: FOS>** `tstkphysaddr`

`"Congratulations!! test [TEST NAME] completed successfully.”`
To ensure the success of a test a congratulations message like this **MUST be appeared without any ERROR messages or PANICs**.

Kernel Heap
Testing

- Dependency Graph:



ALL Required Functions

DON'T FORGET to test each function in MS1 independently in a **FRESH SEPARATE RUN**.

Note:

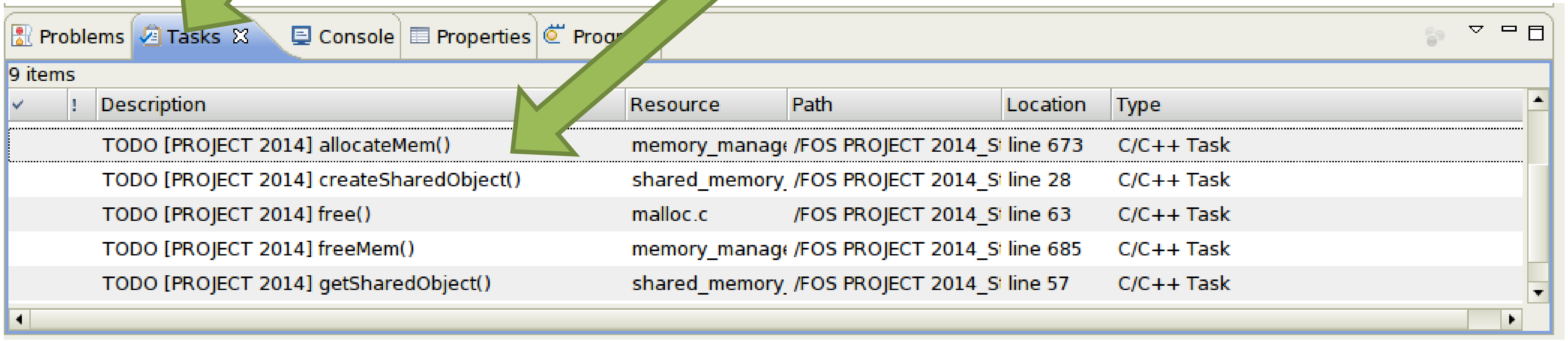
- Those tests to help guide you.
- There are unseen tests will be used in the evaluation so make sure you wrote a correct logic.

Where should I write the Code?

There're shortcut links that direct you to the function definition

[1] Click on "Tasks" Tab

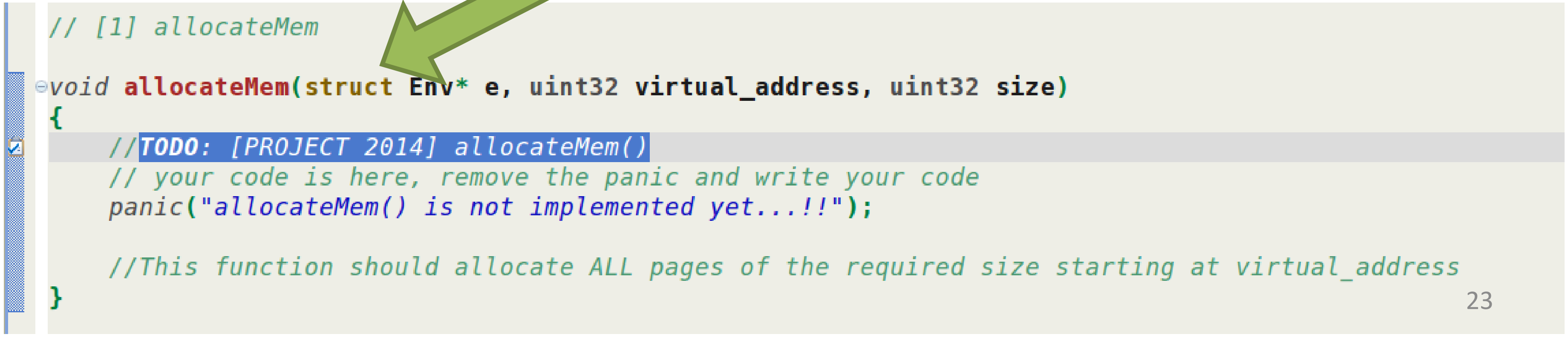
[2] Double Click on the required function



9 items

✓	!	Description	Resource	Path	Location	Type
		TODO [PROJECT 2014] allocateMem()	memory_manag	/FOS PROJECT 2014_S	line 673	C/C++ Task
		TODO [PROJECT 2014] createSharedObject()	shared_memory_	/FOS PROJECT 2014_S	line 28	C/C++ Task
		TODO [PROJECT 2014] free()	malloc.c	/FOS PROJECT 2014_S	line 63	C/C++ Task
		TODO [PROJECT 2014] freeMem()	memory_manag	/FOS PROJECT 2014_S	line 685	C/C++ Task
		TODO [PROJECT 2014] getSharedObject()	shared_memory_	/FOS PROJECT 2014_S	line 57	C/C++ Task

[3] Function body, at which you should write the code

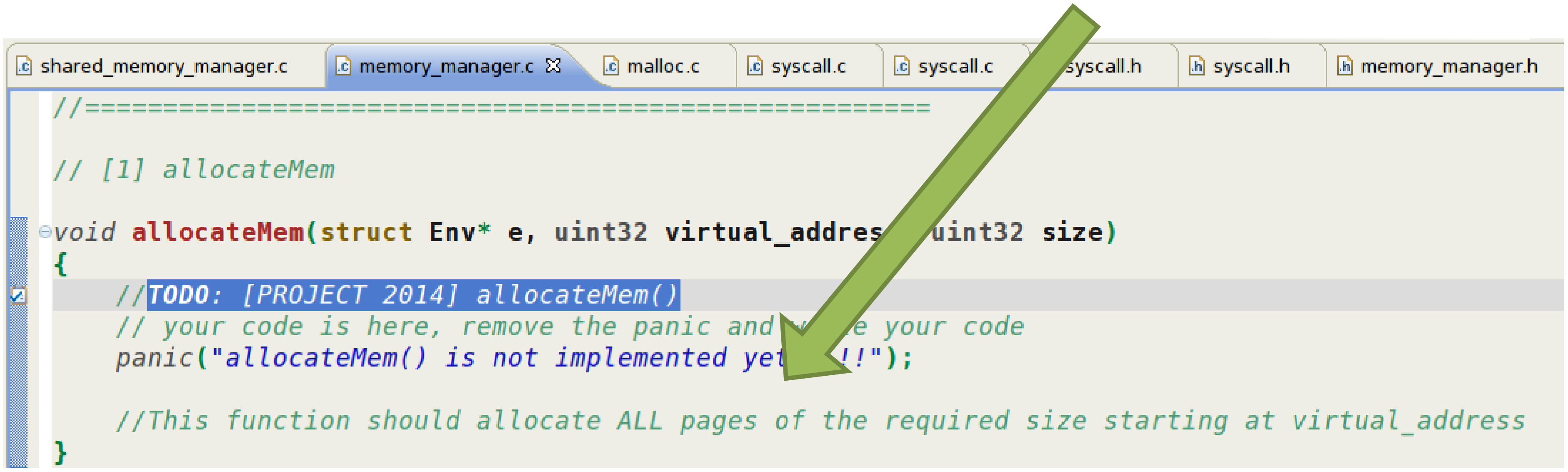


```
// [1] allocateMem

void allocateMem(struct Env* e, uint32 virtual_address, uint32 size)
{
    //TODO: [PROJECT 2014] allocateMem()
    // your code is here, remove the panic and write your code
    panic("allocateMem() is not implemented yet...!!");

    //This function should allocate ALL pages of the required size starting at virtual_address
}
```

What about the steps?

You'll find it inside each function

Detailed Steps



```
//=========================================================

// [1] allocateMem

void allocateMem(struct Env* e, uint32 virtual_addres  uint32 size)
{
    //TODO: [PROJECT 2014] allocateMem()
    // your code is here, remove the panic and w   e your code
    panic("allocateMem() is not implemented yet   !!");

    //This function should allocate ALL pages of the required size starting at virtual_address
}
```

How to Test Your Code?

(Tests **DON'T** guarantee full correct logic,
You **should** implement the correct logic as explained)

- There're **test programs** that test
  - Each function separately
  - Entire project
- Just run the test program & it tell you if it succeed or not

Helper Functions

- Set of **ready-made functions** are available to help you when writing your solution.
- **Detailed description** can be found in **documentation**

Thank you for your care…

Enjoy **making** your **own FOS** ☺