Welcome to COMP20007 Workshops

0	workshop: format, friends and a new possible enemy
1	Arrays and Linked Lists (Tutorial Q1-Q2)
2	ADT: Stacks, Queues (Q3-Q4)
	10-min break for networking
L	• Programming Environment
A	• C revision with some exercises
B	• intro. to multi-file programming (time permitted)

comp20007

Some of the Friends:

- your class-mates
- me: Anh Vo avo@unimelb.edu.au
- Ed: programming platform, discussion forum
- Internet, including ... ChatGPT

A new possible enemy:

- ChatGPT

COMP20007 focuses on algorithm design and efficiency

Problem

Having a collection of data

Need to search for a specified element

Pseudocode (Algorithm)

Efficiency

Programs (in C)

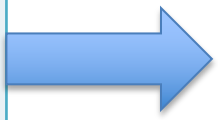
COMP20007 focuses on algorithm design and efficiency

Problem

…

Need to search for a specified element

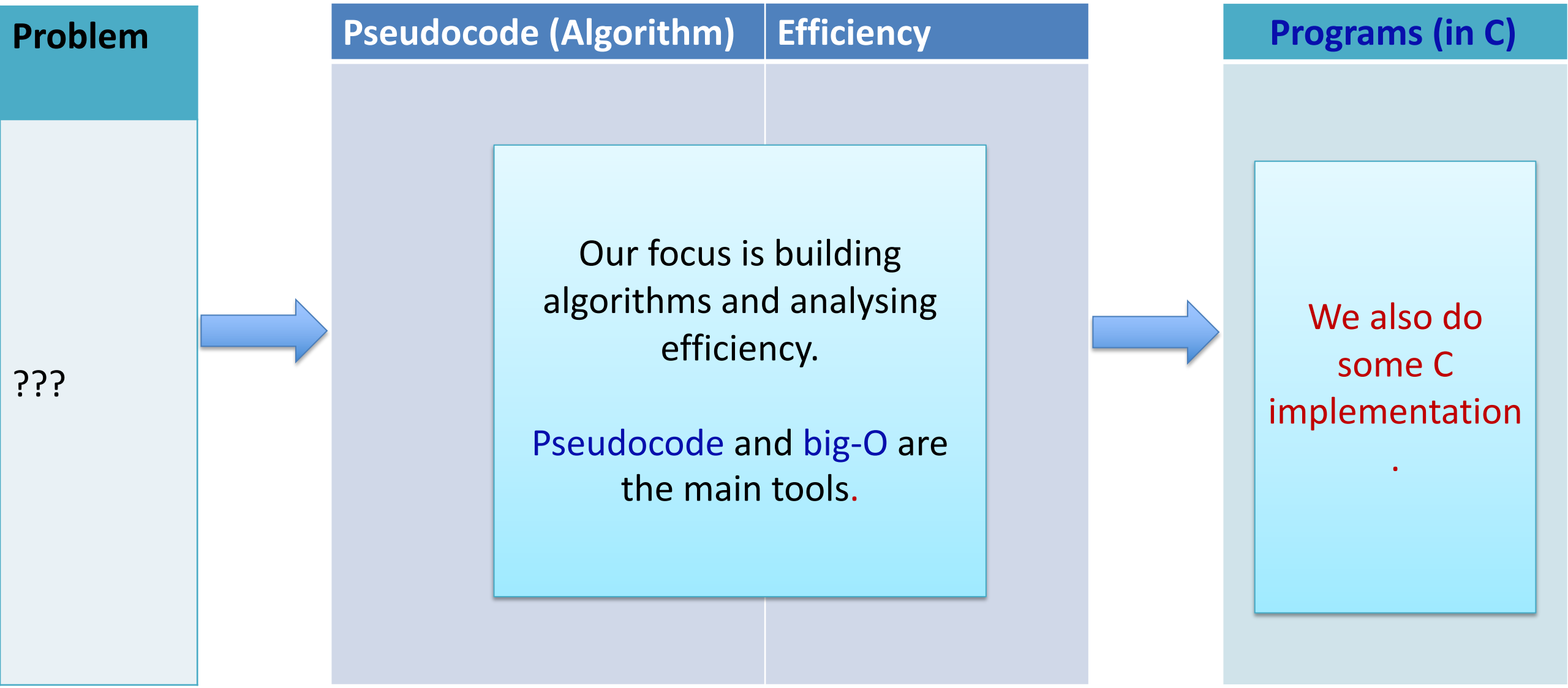
Supposing the collection is static and need a lot of searches



Pseudocode (Algorithm) | **Efficiency**

Programs (in C)

COMP20007 focuses on algorithm design and efficiency



Workshops: Learning-By-Doing

in your own time

W = 1

Attend & Revise lectures of week W

W++

- Prepare with
  Ed → Lessons → Week **W**: Workshop
- Learn from books, **ChatGPT, Google...**

- Finish the outstanding tasks
- Check with the solutions provided

in workshop

Tute time:
- be active in discussions
- pen & papers needed

5 minutes: networking

Lab time:
- do at least one hard task
- make sure you can do others
- be cooperative

Topic 1: Arrays & Linked Lists

- What are they? How to specify?
- What are the main differences?

Q2.1: Arrays

Describe how you could perform the following operations on *sorted* and *unsorted arrays*, and decide if they are *O(1)*, *O(log n)*, or *O(n)*, where *n* is the number of elements initially in the array. Assume that there is no need to change the size of the array to complete each operation.

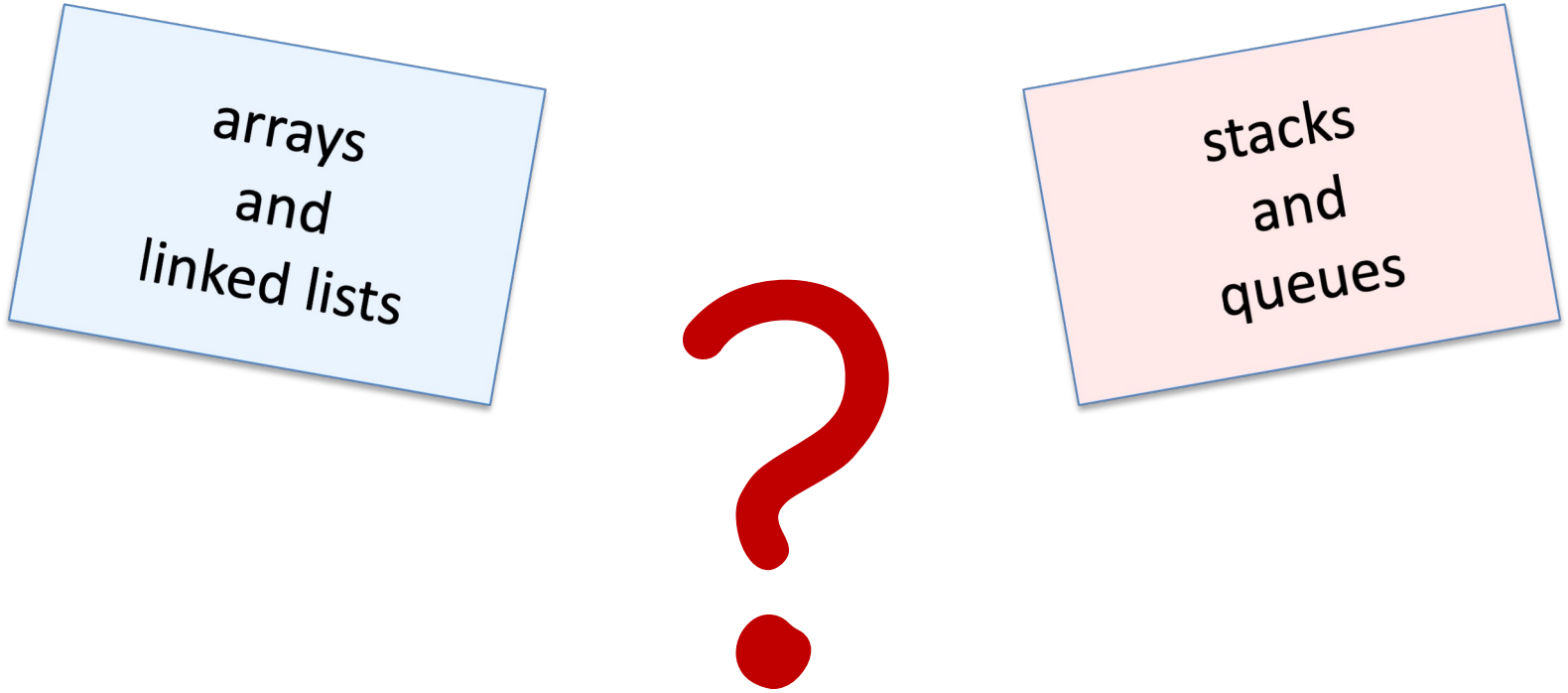
Operation	Unsorted Arrays	Sorted Arrays
Searching for a specified element	**O(n)** • do a linear search	**O( )** •
Inserting a new element	**O( )** •	**O( )** •
Deleting the final element	**O( )** •	**O( )** •
Deleting a specified element	**O( )** •	**O( )** •

Q2.2: Linked Lists

Describe how you could perform the following operations on singly-linked and doubly-linked lists, and decide if they are *O(1)*, *O(log n)*, or *O(n)*, where *n* is the number of elements initially in the linked list. Assume that the lists need to keep track of their final element.

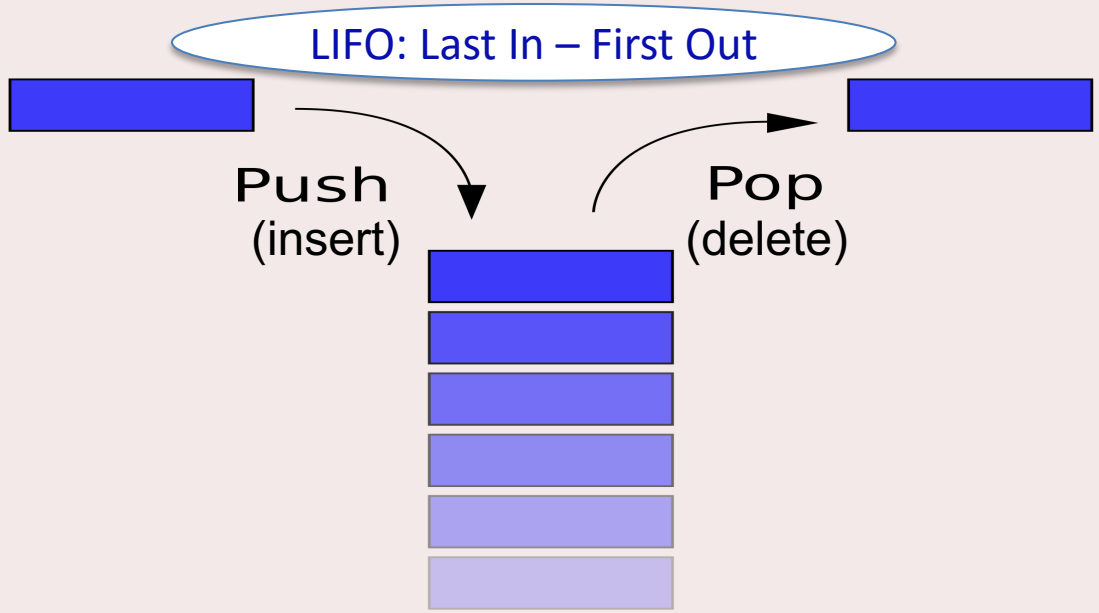
Operation	Singly	Doubly
Inserting a node at the start	O( )	
Inserting a node at the end		
Deleting the first node (at the start)		
Deleting last node (at the end)		

- Compare



An Abstract Data Type (ADT): Stack (LIFO)

http://www.123rf.com/stock-photo/tyre.html



adapted from https://simple.wikipedia.org/wiki/Stack_(data_structure)

Stack Operations

`push(x)` : insert element `x` to (the top of) stack

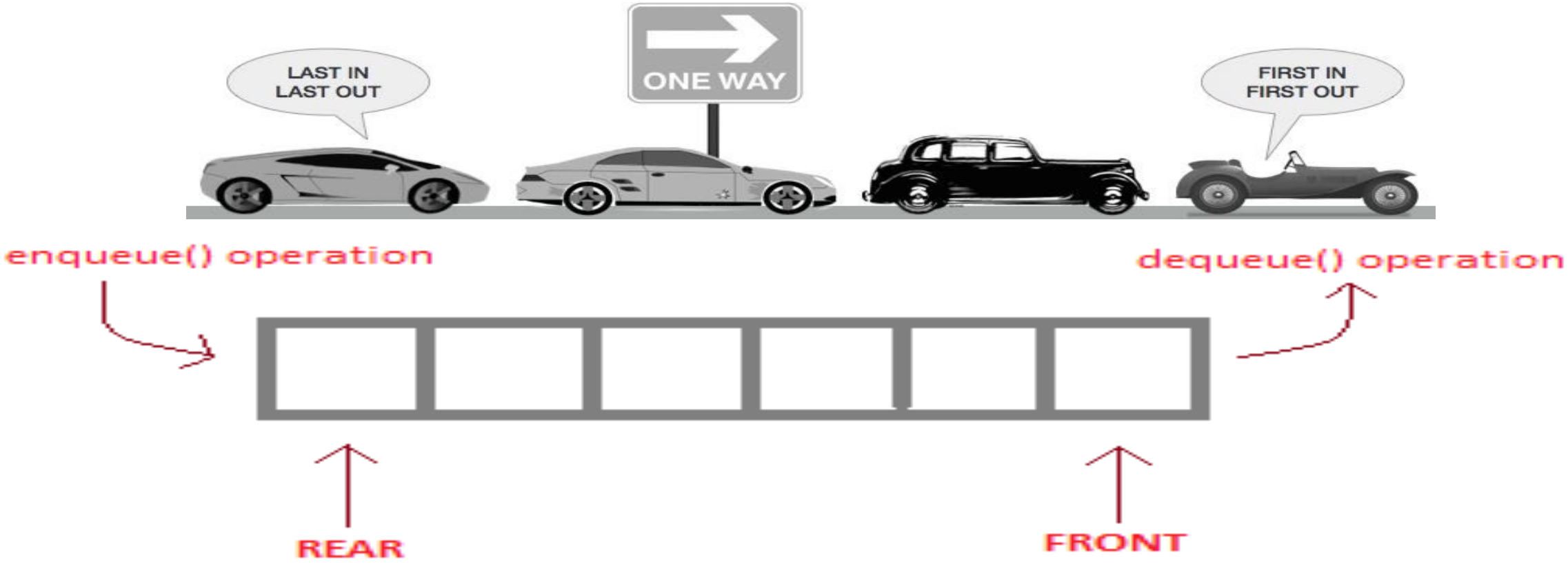
`pop()` : remove and return an element from (the top of) stack

`isEmpty()` : check if stack is empty

`create()` : create a new, empty stack

`delete()` : delete (free all associated memory)

Another ADT: Queue (FIFO)



Queue Operations	**`enqueue(x)`**: add **`x`** to (the rear of) the queue **`dequeue()`**: remove and return the element from (the front of) the queue **`create()`**: create a new, empty queue **`isEmpty()`**: check if queue is empty **`delete()`**: delete a queue (free all associated memory)

Q2.3: Stacks

Describe how to implement `push` and `pop` using

- an unsorted array?
- a singly-linked list?

Using an (unsorted) array	Using a (singly-)linked list

Q2.4: Queues

Describe how to implement **`enqueue`** and **`dequeue`** using an unsorted array, and using a singly-linked list. Is it possible to perform each operation in constant time?

Using an array	Using a linked list

Q2.5 [homework]: Stacks & Queues

If you have access only to stacks and stack operations, can you faithfully implement a queue? How about the other way around?

using stacks to implement a queue

enqueue	dequeue

using queues to implement a stack

push	pop

10-minute break

stretch exercises

networking

Just for fun (try now or at home)

google ”algorithm for making friends” and watch ”The Friendship Algorithm” (a 2.5-minute videos).

Lab Time: Use Ed for exercises and assignments

1. *Start* with `helloworld.c` [DoItTogether with Anh]
2. (Together) Implement functions in `functions.c`, which reviews *function and function parameters*
3. *dynamically resizing arrays* with `malloc`/`calloc` and `free`. Forgot `malloc`? Try command "`man malloc`" in your terminal.

Why `Ed`?

- Strong: powerful editor, shell, compilers, `valgrind`, `gdb`, …
- Safe : codes and files will never be lost
- Sound : codes/files can be accessed from any devices
- Sane : your assignments will be tested on `Ed`

4. (time permitted): break the resizing array code (Problem 3) into a few files and compile together

Wrap Up

array and linked list as concrete data types.

stack and queue as Abstract Data Types (ADT):

- operations,
- implementation using array and linked list.

C revision, especially:

- functions, pointers, `malloc`/`realloc`, `free;`
- dynamically resizing arrays (or just *dynamic arrays*).

Technical stuffs:

- Use Ed for programming exercises & assignments!
- Self-Learn to use `Ed`'s `gcc` and debugging tools `gdb`, `valgrind` at home.

Have Fun with comp20007 and avo

Additional Pages

Data Types & ADT

A *concrete data type*, such as array or linked list, specifies a representation of data, and programmers can rely on that to implement operations (such as `insert`, `delete`).

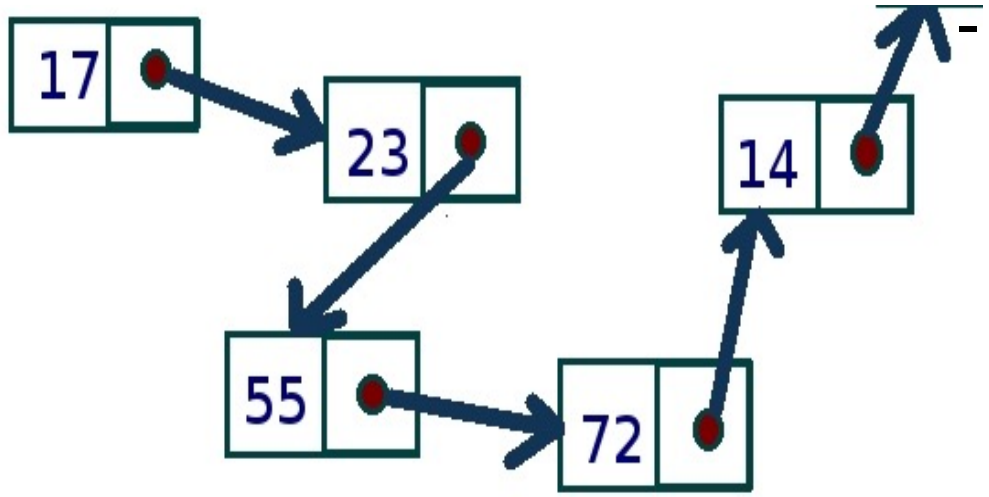
An *abstract data type* specifies possible operations, but not representation. Examples: stacks, queues, dictionaries.

- When implementing an ADT, programmers use a concrete data type. For example, we might attempt to employ array to implement stack.
- When using an ADT, programmers just use its facilities and ignore the actual representation and the underlined concrete data type.

Two concrete data types: Arrays & Linked Lists

Access A[k] in O(1) time!

Access L[k] in O(n) time!



In C:

- How to specify an array? How to traverse it?
- How to specify a linked list? How to traverse it?

Example of using Stacks & Queues ?

Stack is widely used in implementation of programming systems. For example, compilers employ stacks for keeping track of function calls and execution.

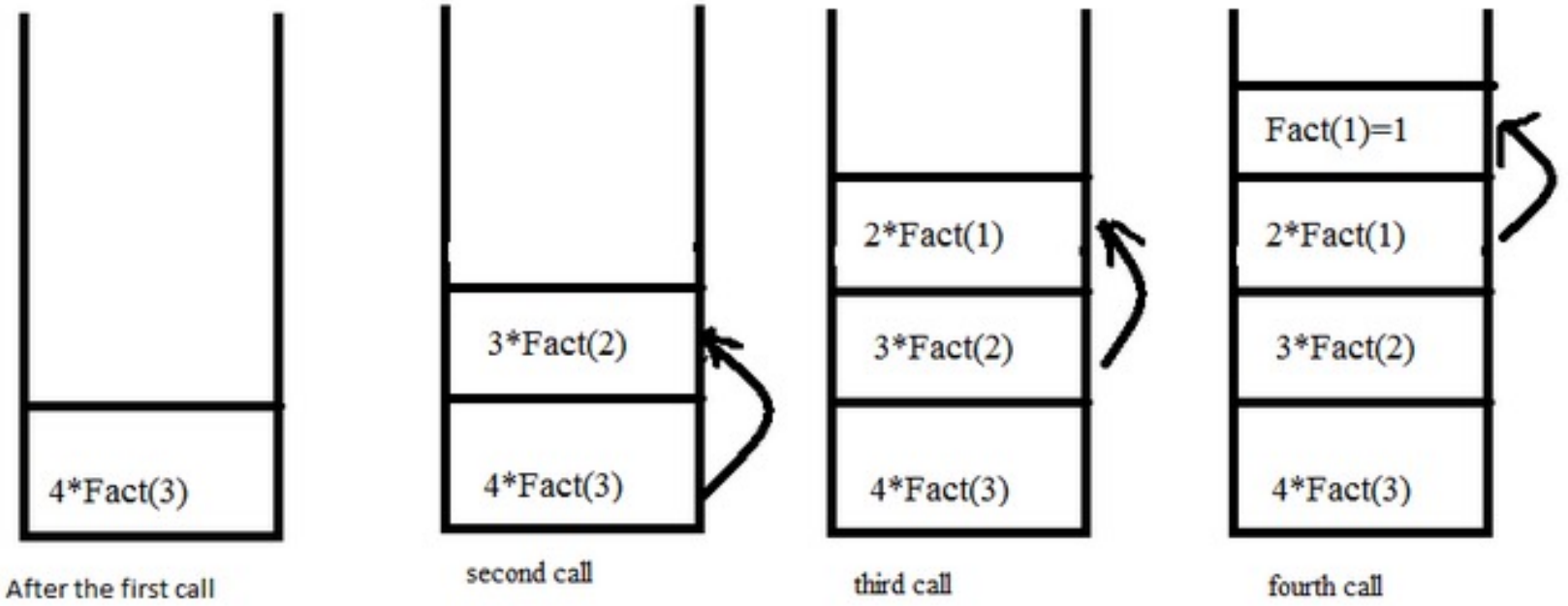
Stack for :

Fact(4)

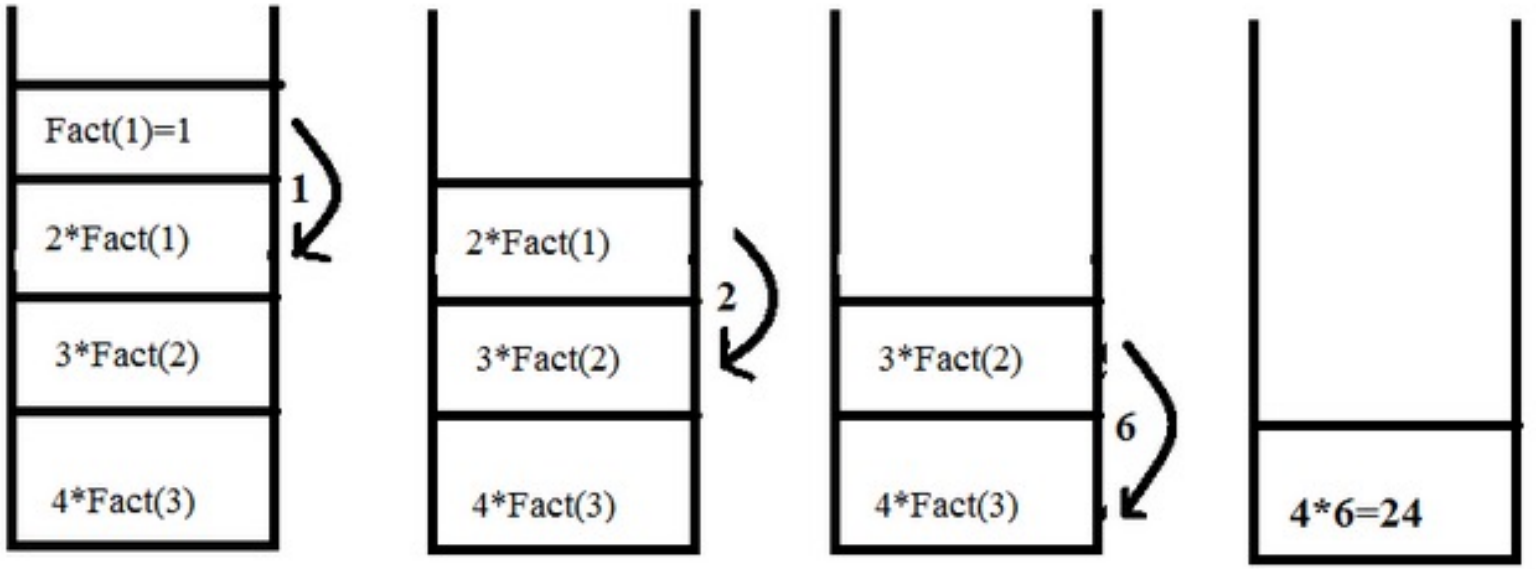
```
int Fact( int n ) {
  if ( n<=1 )
      return 1;
  return n*fact(n-1);
```

Image source: http://stackoverflow.com/questions/19865503/can-recursion-be-named-as-a-simple-function-call

When function call happens previous variables gets stored in stack



Returning values from base case to caller function



ADT: Queue (FIFO)