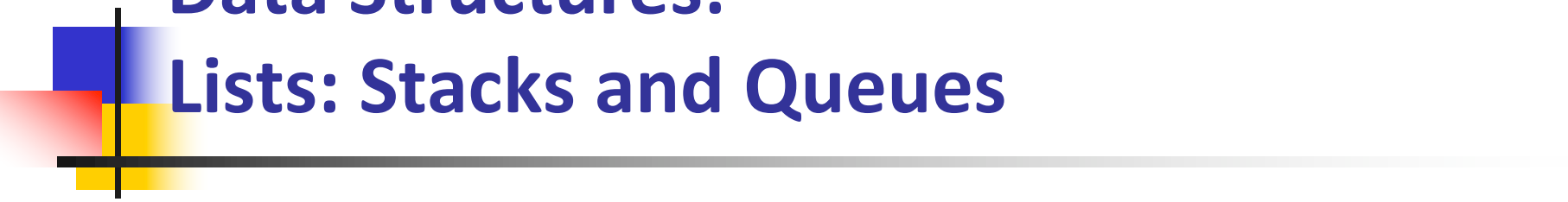


Data Structures: Lists: Stacks and Queues

Won Kim
(Lecture by Youngmin Oh)
Spring 2022

Course Objectives

- Learn the Concepts of Fundamental Data Structures.
  - used in processing data using computers
- Learn How the Concepts Are Applied.
- Learn How to Map the Concepts to Computer Programs in C.

Course Contents

- 8-10 Lectures
- 6-8 Labs (in-class)
- 4-8 Homeworks
- 2 Exams

** Exact numbers of each assignment are subject to change*

Course Grading Policy

- Exams 40
  - mid-term: 20
  - final: 20
- Homework: 25
- Labs & Quizzes: 20
- Attendance 15

Total: 100

Textbook

- No Textbook
- Reference
  - Fundamentals of Data Structures in C: 2nd Edition
    - Ellis Horowitz, Sartaj Sahni, Susan Anderson-Freed (W.H. Freeman and Company)
    - Widely used, programming exercises, in English

Data Structures

- **Techniques for Organizing and Storing Data**
- **Why?**
  - To Allow Fast Access for Specific Purposes
  - "Access" means Read, Update, Delete, Copy, Move
- Many Possible Purposes -> Many Possible Techniques

What This Course Will Cover

- Main Memory Data Structures
  - lists
  - trees
  - graphs
  - hashing
- Secondary Storage Data Structures
  - trees
  - hashing
- Algorithms
  - sorting
  - tree operations
  - graph traversal

Issue of "Scale"

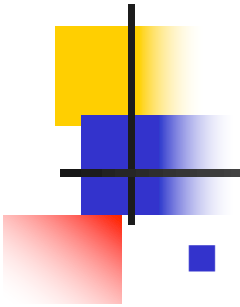
- Data Structures Are Important When There Are Lots of Data To Store and Access.
- The Following Questions are Meaningless.
  - What is the best data structure for (20, 15, 11, 35)?
  - What is the best data structure for (John, Mary, Paul, Nancy, Peter)?

Applications and Tradeoffs

- There Are Useful Applications for Almost Each Well-Developed Data Structure.
- There Are Tradeoffs Between Any Pair of Data Structures.
  - Almost every data structure has advantages and disadvantages.
  - There is no “best data structure for every purpose”.

Lists

- Each of the following is a list of data items. Each has different uses and different properties.
  - Arrays
  - Stacks
  - Queues
  - Linked Lists



Fundamental Data Structures

- Of the four types of list, arrays and linked lists (and structs) are basic data structures.
- All other data structures make use of them.

Stack and Queue

Stack: Applications

- Tree data structures (to learn in this course)
- Binary expression evaluation (in a compiler)
- System Stack in OS
  - Activation records
    - nested function calls, including recursive function calls

Implementing a Stack

- Using an Array (global or local)
  - non-circular buffer
  - circular buffer
- Using a Linked List

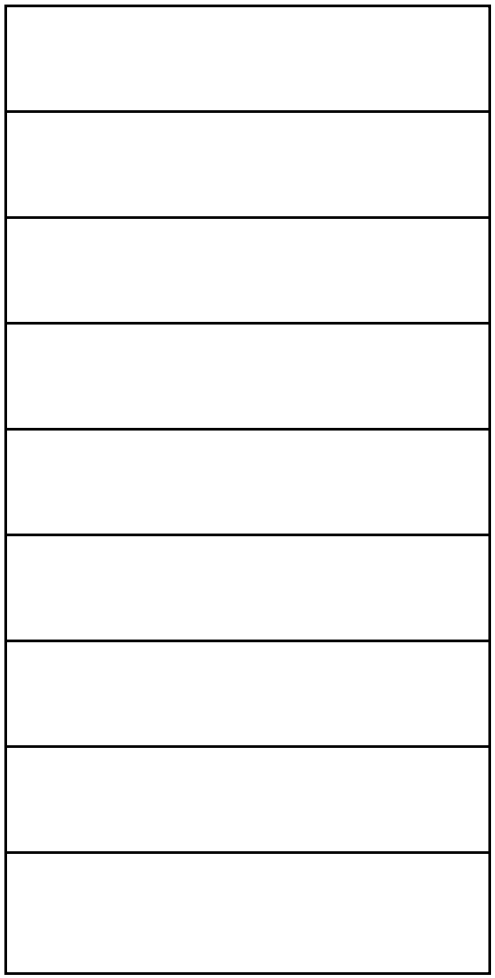
Using a Non-Circular Buffer

- One-Dimensional Array
- (datatype) stack[stack_size]
- (ex.) char stack[100]

- Variable “Top”
- initially top = -1 (empty stack)

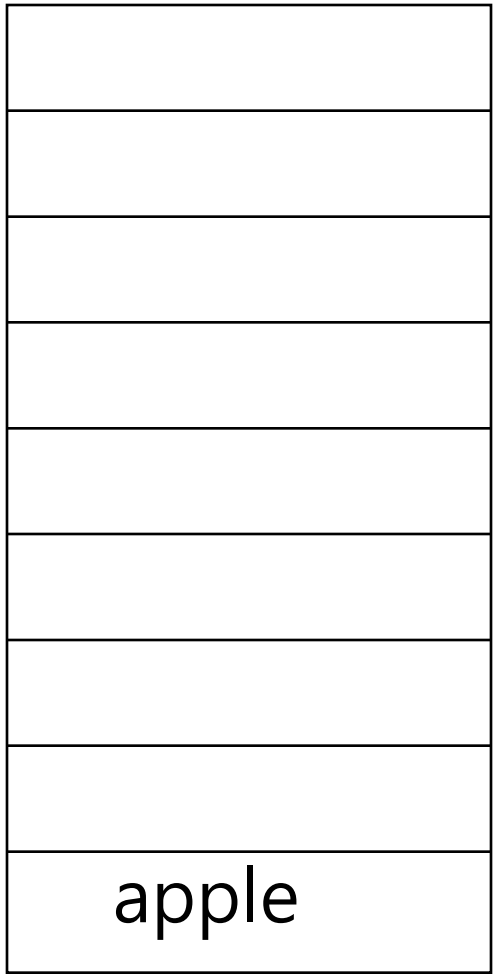
- insert(element) or push(element),
  delete() or pop(),
  stack_full(), stack_empty()

Stack Implementation (Using an Array) (1/12)



top=-1

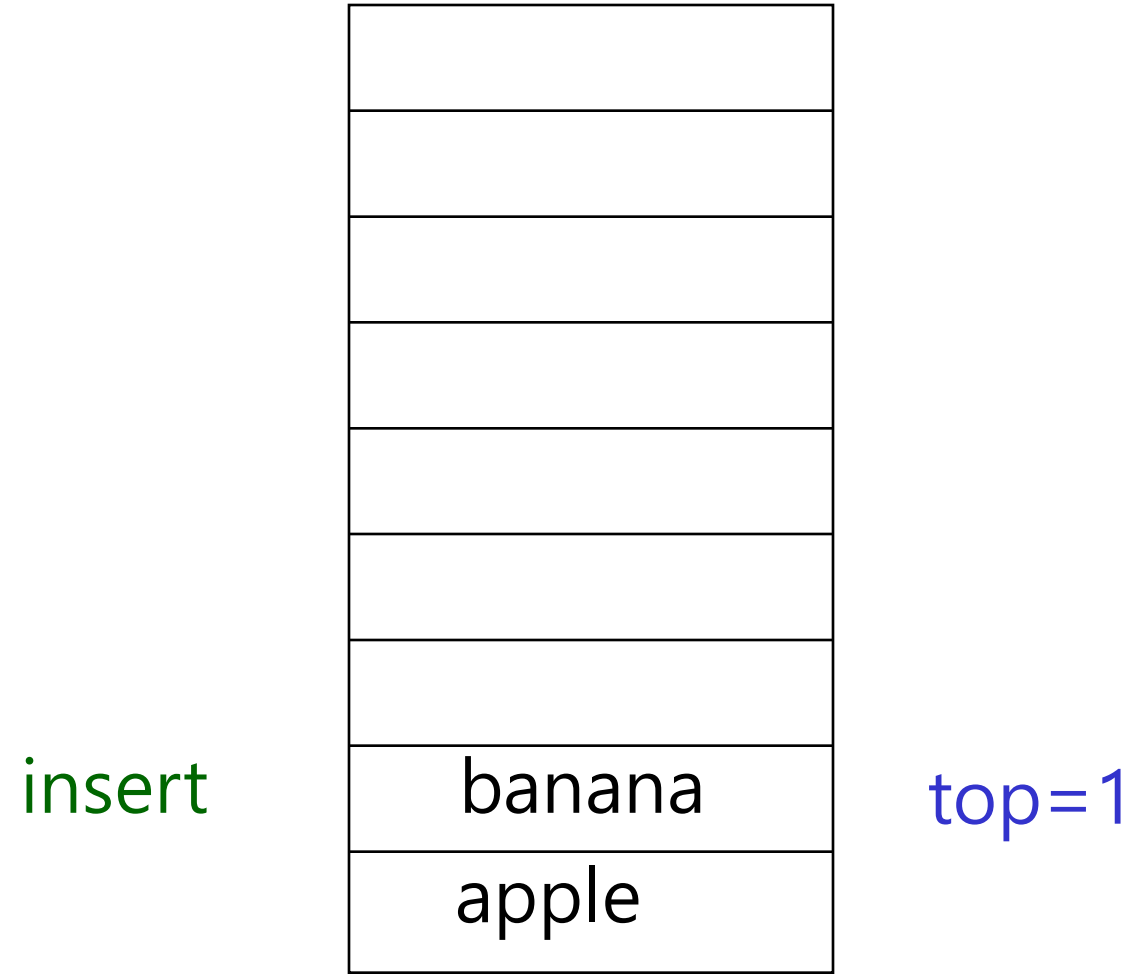
(2/12)



insert

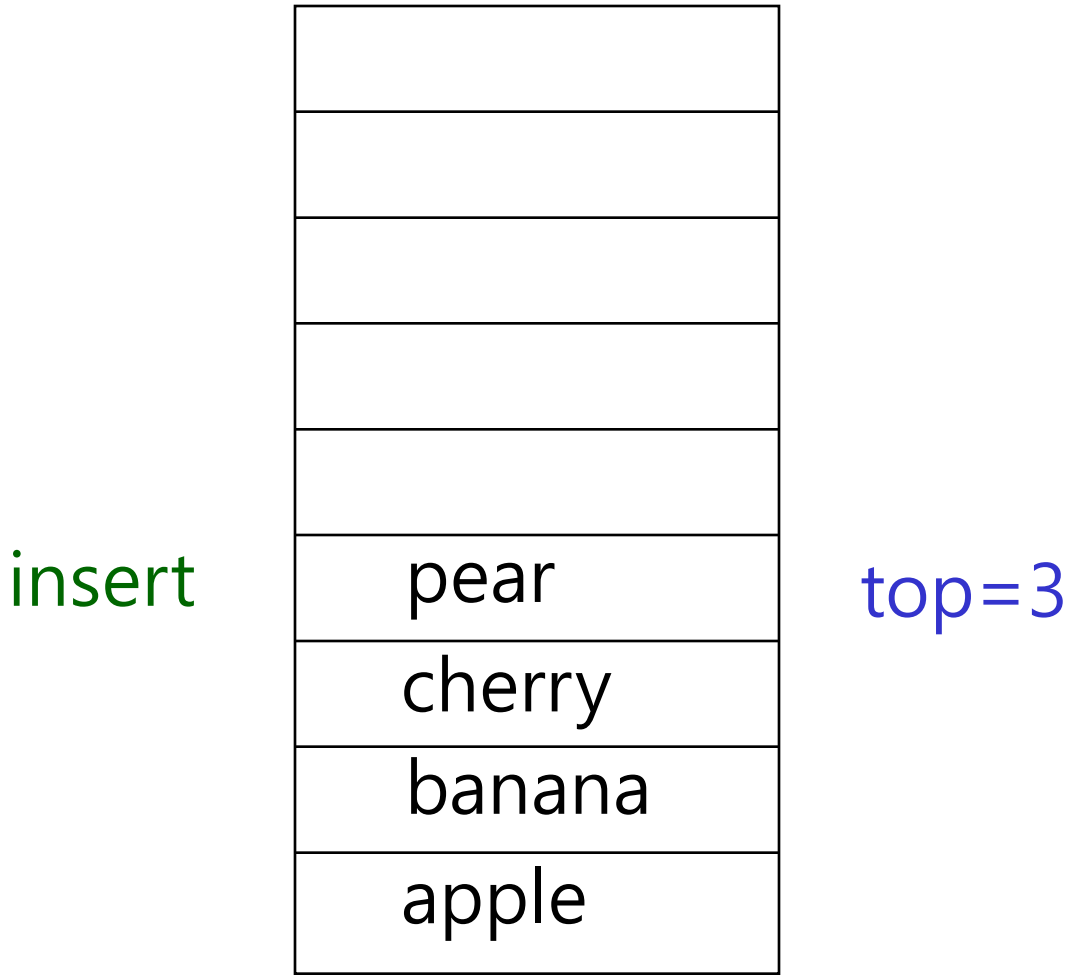
top=0

(3/12)

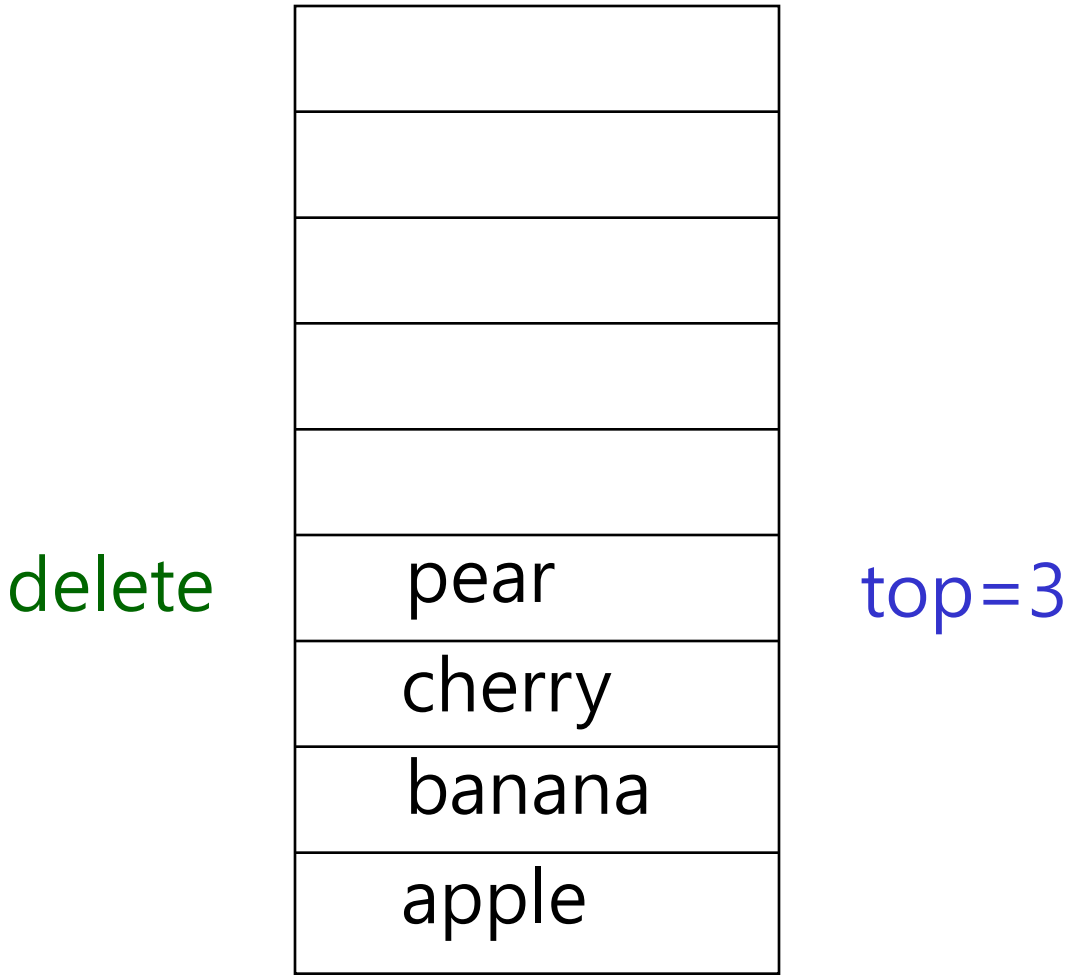


(4/12)

(5/12)



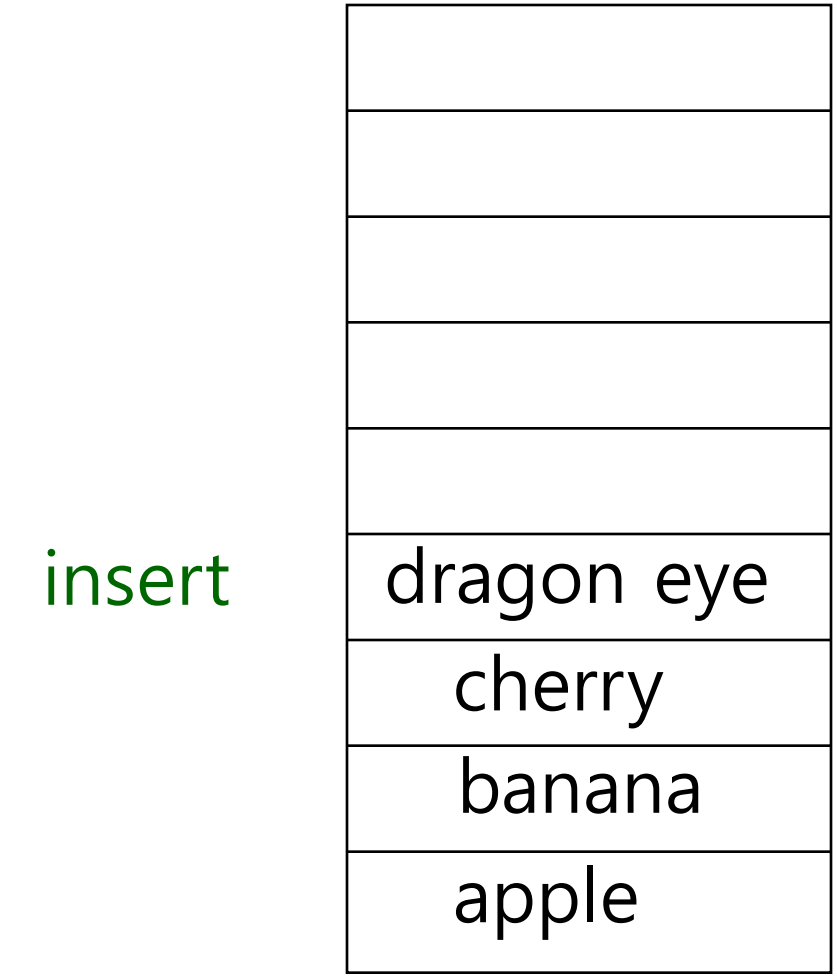
(6/12)



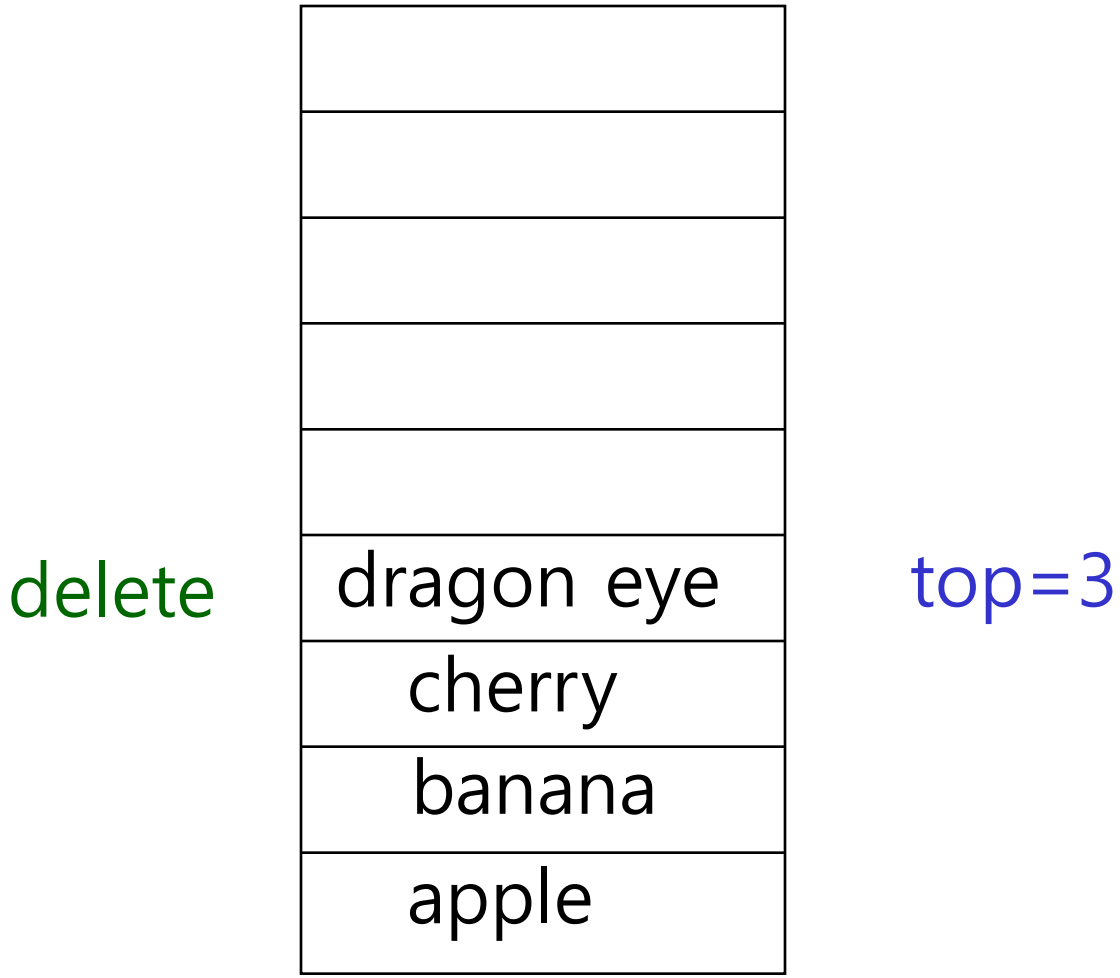
(7/12)

top=2

(8/12)



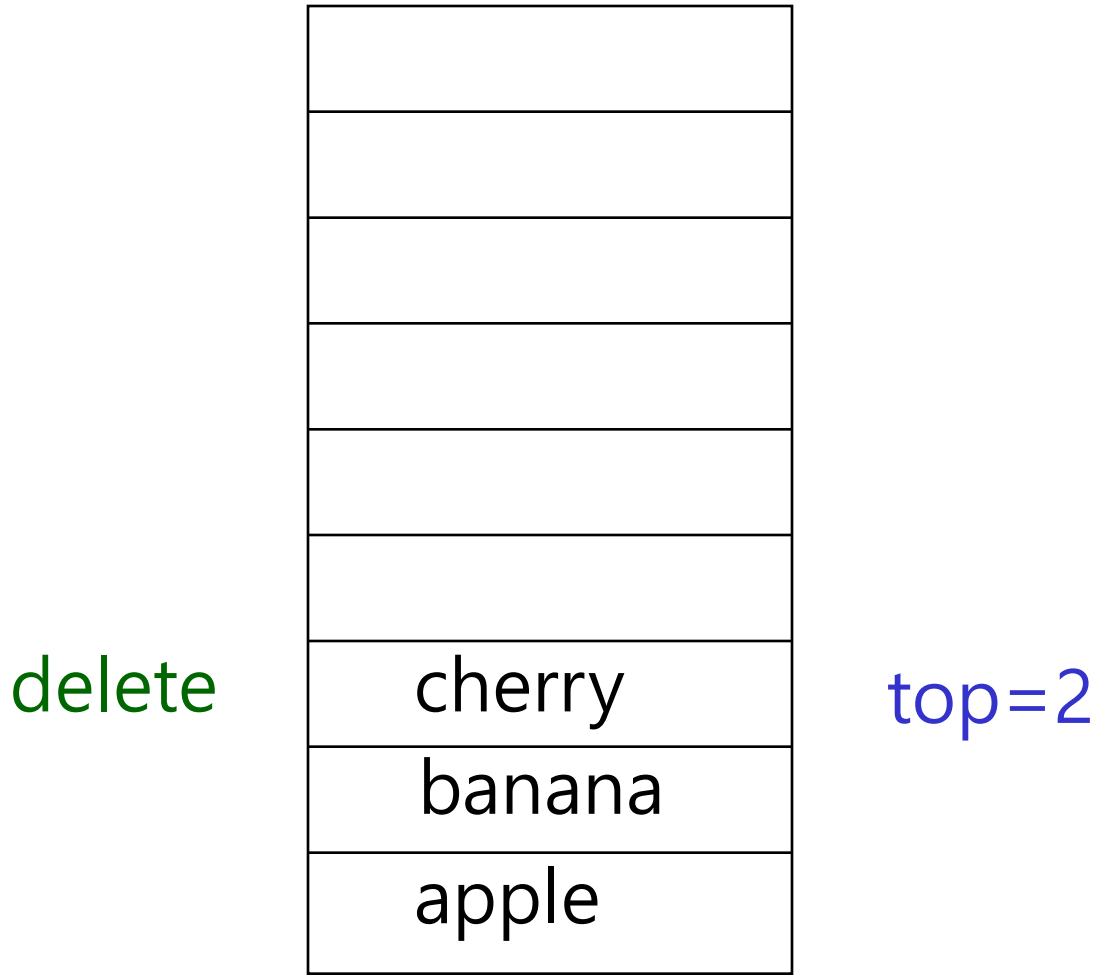
(9/12)



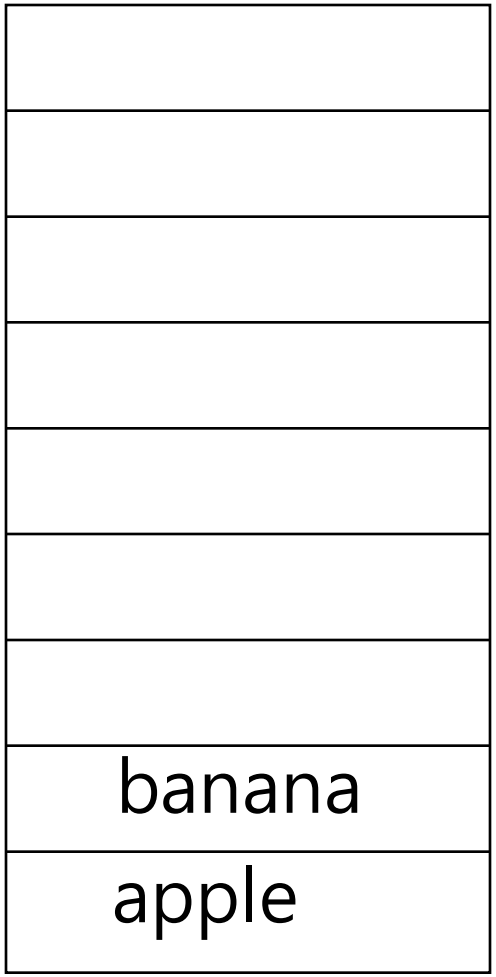
(10/12)

top=2

(11/12)



(12/12)



top=1

Queue

Implementing a Queue

- Using an Array (global or local)
  - non-circular buffer
  - circular buffer
- Using a Linked List

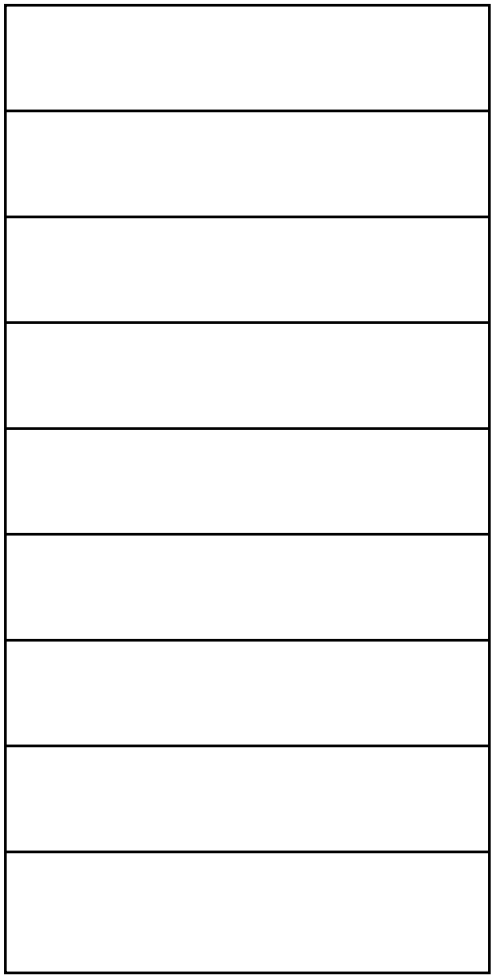
Using a Non-Circular Buffer

- One-Dimensional Array
- (datatype) queue[queue_size]

- Variable “Front”
- Variable “Rear”
- initially front = rear = -1 (empty queue)

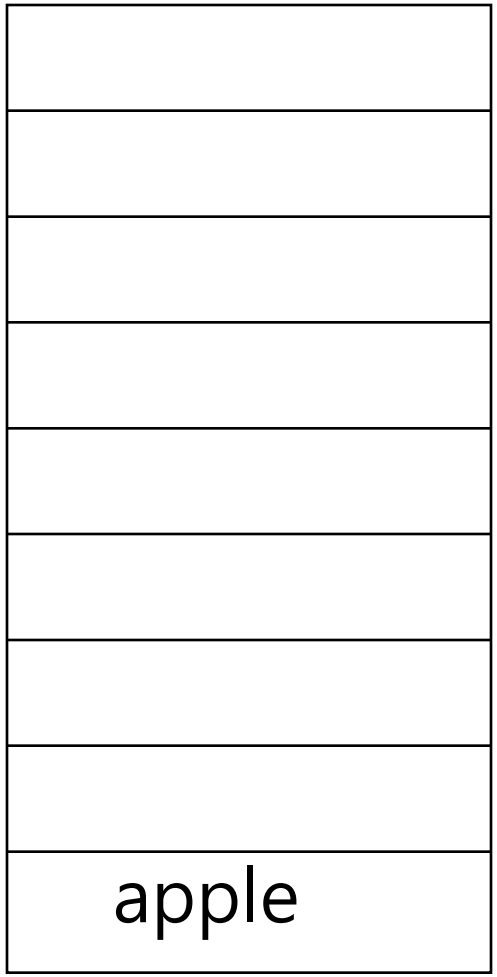
- insert(element) or enqueue(element), delete() or dequeue(), queue_full(), queue_empty()

Queue Implementation (Using an Array) (1/12)



front=-1 rear=-1

(2/12)

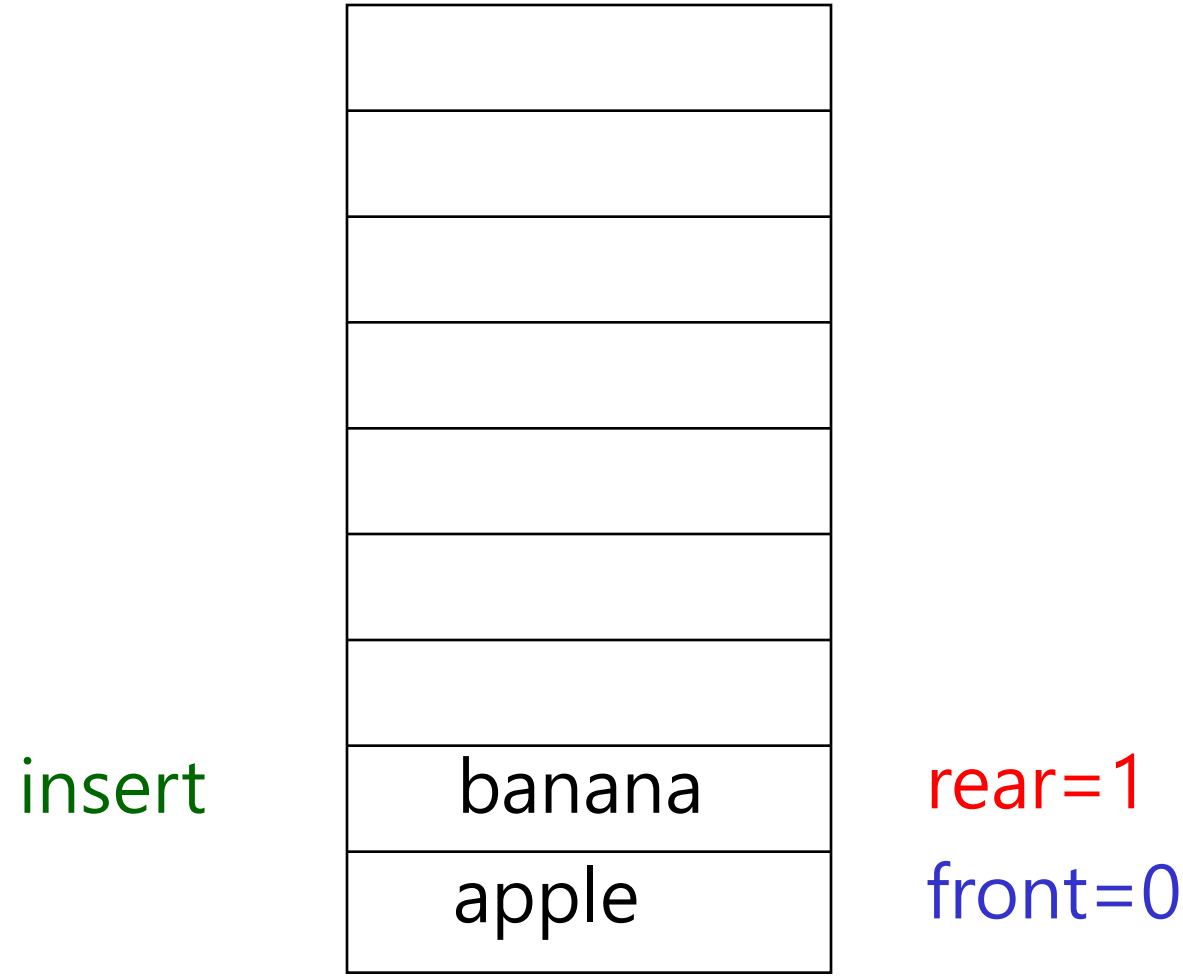


insert

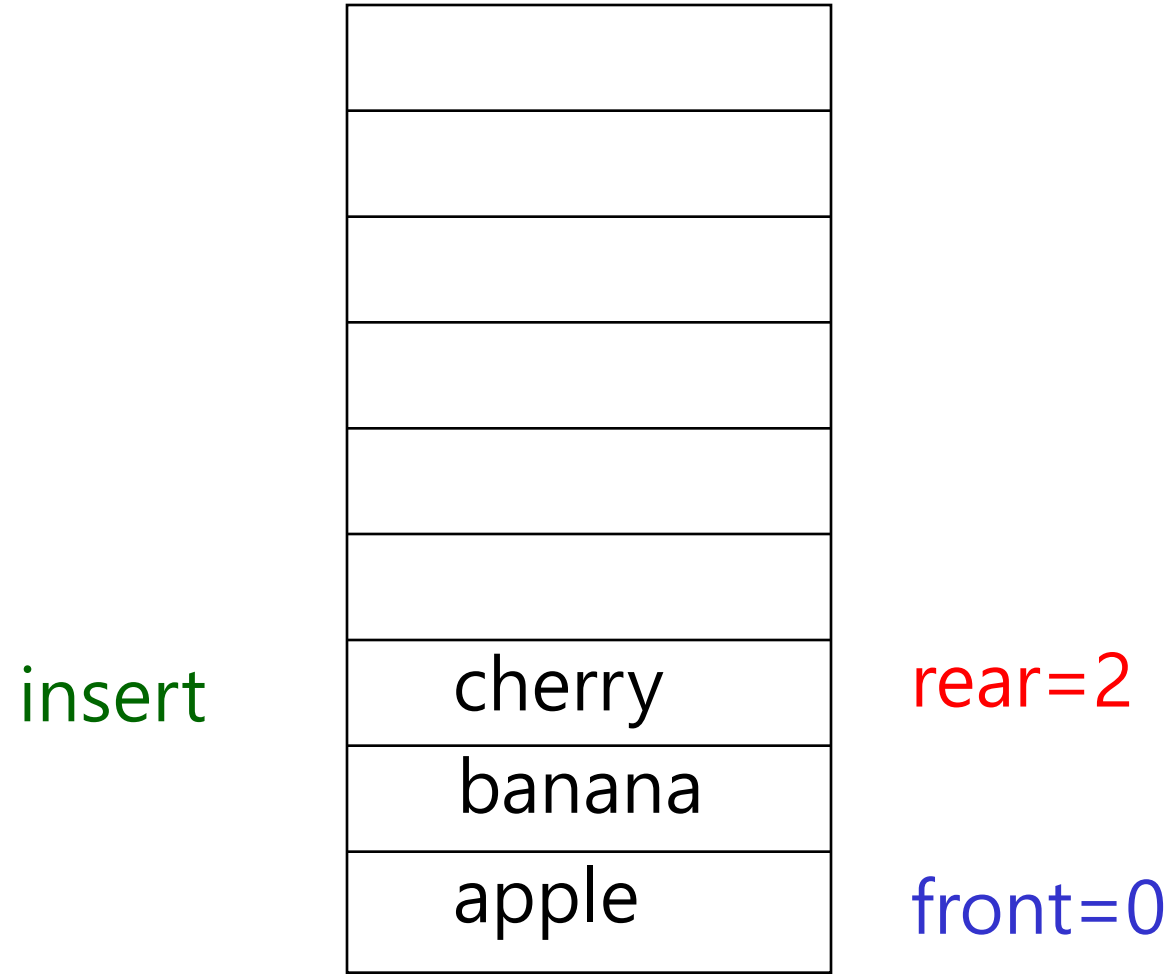
front=0

rear=0

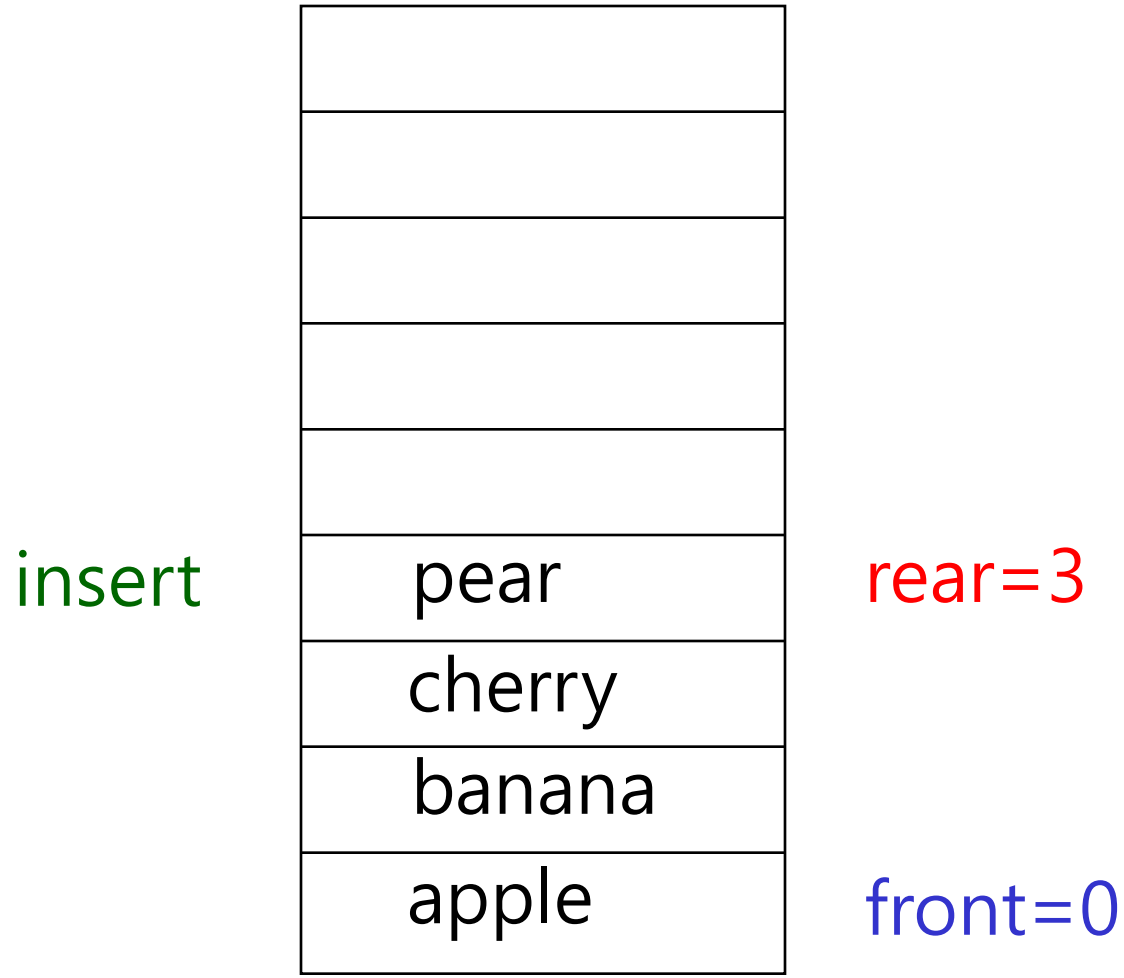
(3/12)



(4/12)

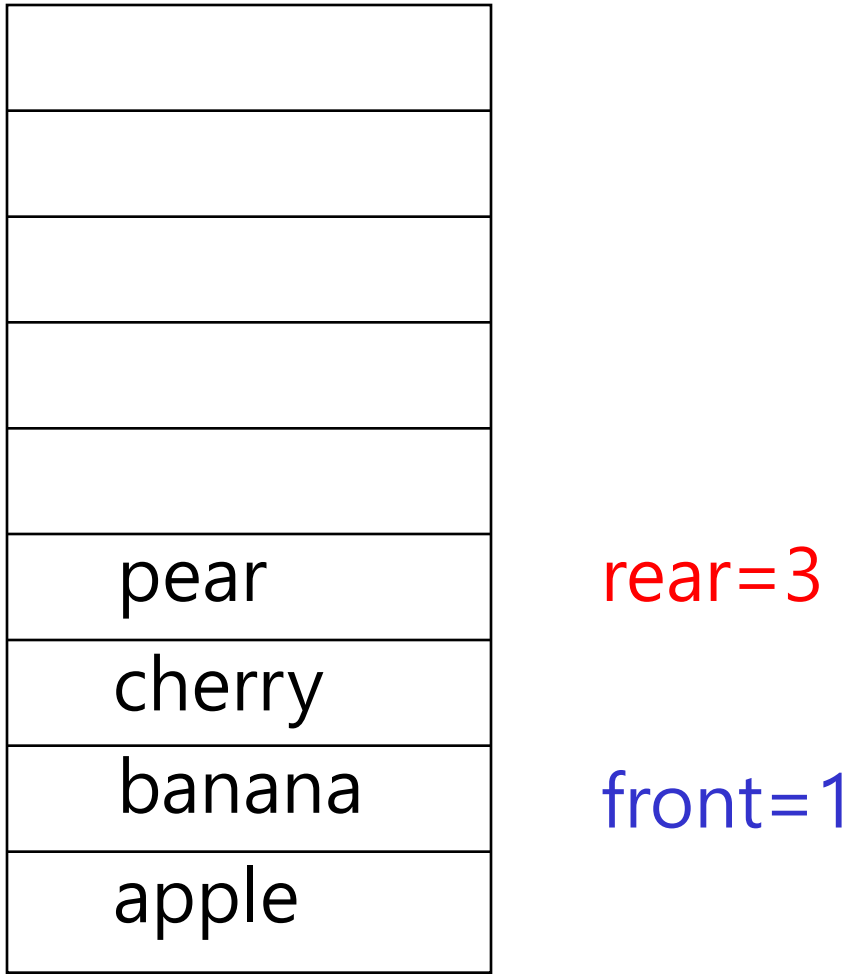


(5/12)

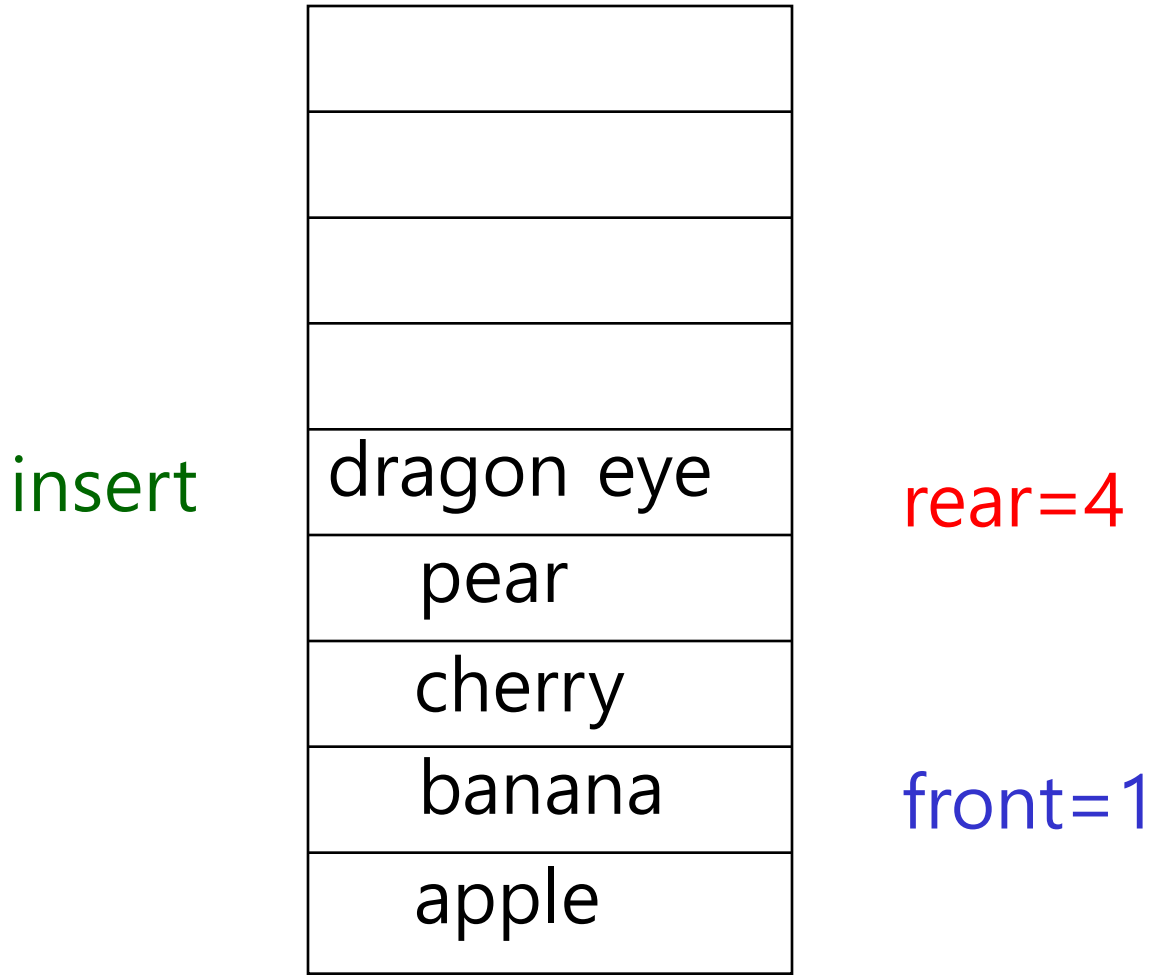


(6/12)

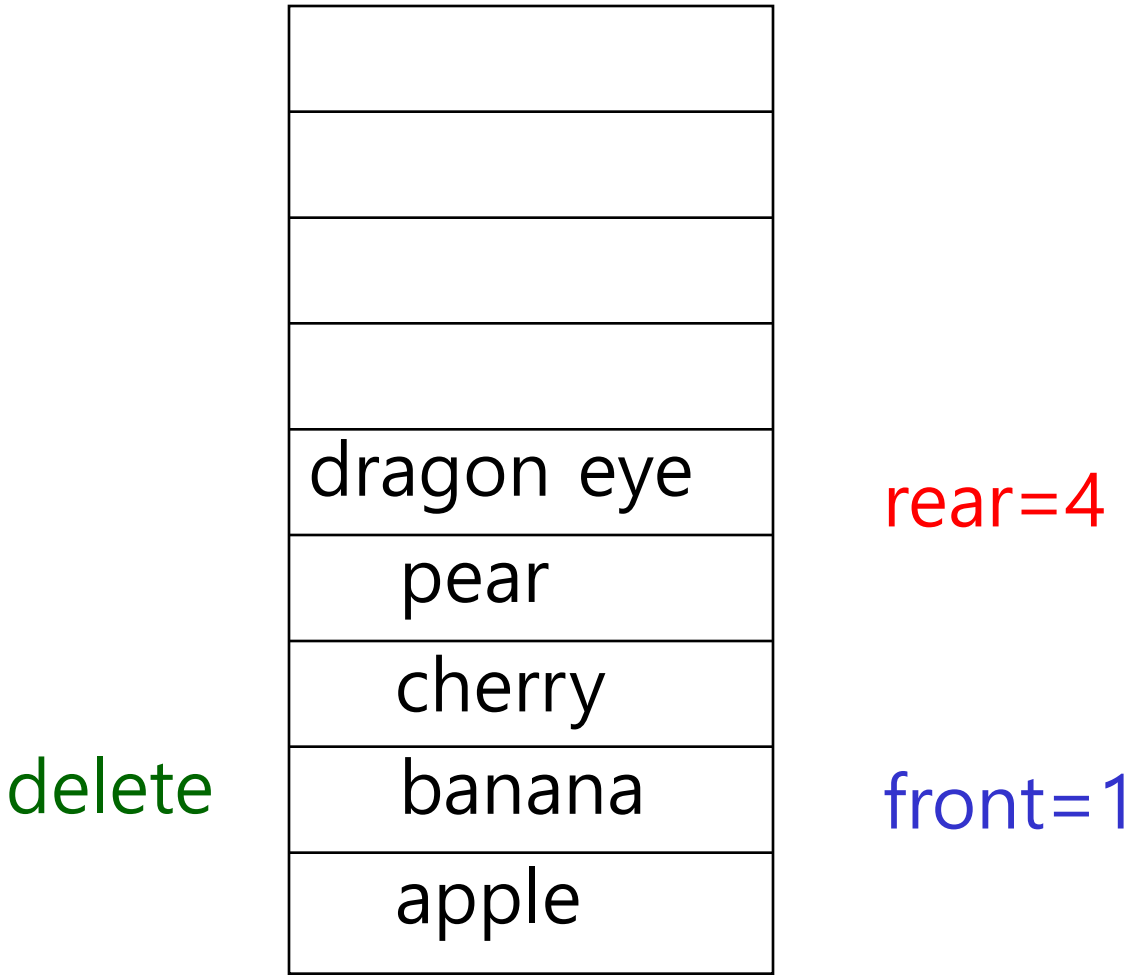
(7/12)



(8/12)



(9/12)



(10/12)

(11/12)


dragon eye
pear
cherry
banana
apple
rear=4
front=2
garbage
garbage

(12/12) wasted space !!

How to Reuse Space?

Lab 1

Software Development Process

- Understand All the Requirements
- Plan
  - Development, Testing, Documentation
- Basic Design
- Implement
  - detailed design, code
  - test (code review, test suite)
- Document

Principles of Good Coding

- Follow All the Requirements
- Design a Good Structure
  - divide work into independent and reusable functions
- Make It Easy to Read
  - structure, (variable, function) naming, layout (spacing)
  - function (and inline) comments
- Make It Efficient
  - minimum (instructions, CPU time, memory use)
- Make It Error-Free
  - defensive coding (check for errors)

Principles of Good Testing

- Check All the Requirements
- Do Manual Code Inspection
  - (same as checking PPT, report, exam answers before submitting)
- Create a Test Plan
  - test scenarios (e.g., sequence of push and pop)
  - test environment (e.g., reduce the data structure size – if array size is 1000, for test purpose, set it to 10)
- Create a Test Suite
  - test cases, and golden (correct) result set
- Document and Save the Test Plan and Test Suite

Lab 1-1

- Implement a Stack Program for a (non-Circular) Integer Stack of size 10
- 4 functions, using an array of size 10
  - push (int)
  - int pop ()
  - int stack_full ()
  - int stack_empty ()
- Test the Stack Program
  - Write the main function to exercise the 4 functions

Implementing Stack Operations

- Do not use pointers to call functions
- (for testing) Use scanf, printf only in "main"
- Use defensive coding
  - push
    - call stack_full before "push"
  - pop
    - call stack_empty before "pop"

Function Comments

- push
  - description: appends data to the stack
  - input: data to append (the stack is a global structure)
  - output: none
- pop
  - description: removes data from the stack
  - input: none
  - output: data on top of the stack

Lab 1-2

- Implement a Queue Program for a (non-Circular) Integer Queue of size 10
- 4 functions, using a global array of size 10
  - enqueue (int)
  - int dequeue ()
  - int queue_full ()
  - int queue_empty ()
- Test the Queue Program
  - Write the main function to exercise the 4 functions

Notes About Point Deductions

- Even if the code runs, points will be deducted for
  - inadequate comments
  - not following the spec
  - poor program structure
  - poor readability of the result screen
  - needless renaming of such standard terms as “push”, “pop”, “front”, “rear”, etc.

End of Class