SFBU **Name:** _____

Practical Applications of Algorithms-Spring 2024

Final-Review

Time Limit: 75 Minutes

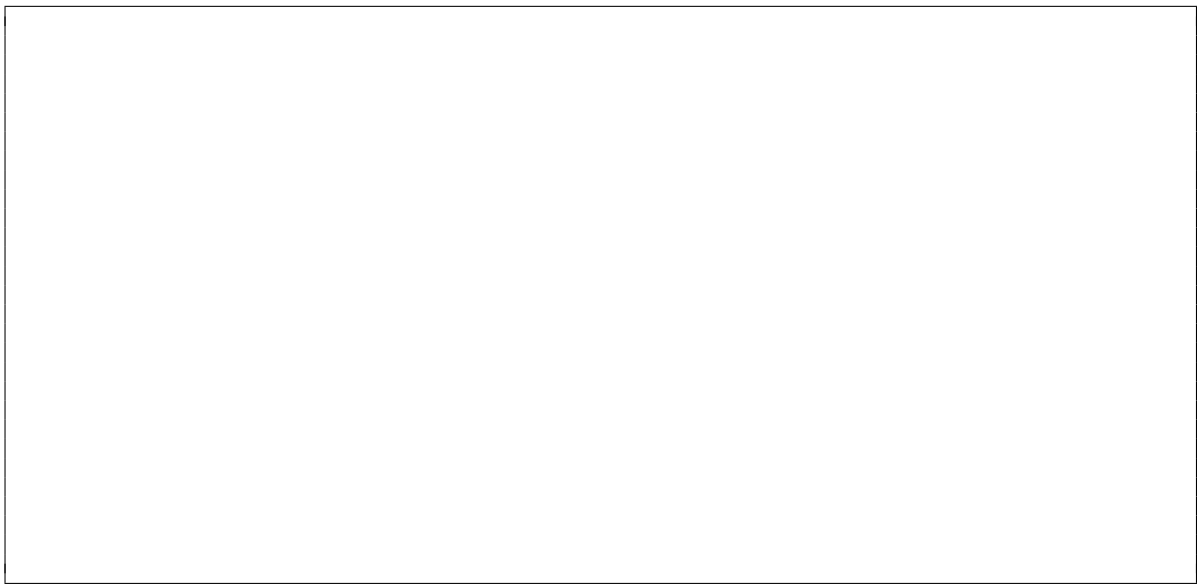
---

1. This exam contains 32 pages (including this cover page) and 18 questions.
2. Total of points is 180.
3. Answer all questions. The marks for each question are indicated at the beginning of each question.
4. This **IS a CLOSED BOOK** exam.
5. Smart phones and watches must be kept in your bags during the test.

---

1. (10 points) "Remove Element". Provide the code for the following question.

```
Given an integer array nums and an integer val, remove all occurrences of val
in nums in-place. The order of the elements may be changed. Then return the
number of elements in nums which are not equal to val.
Consider the number of elements in nums which are not equal to val be k,
to get accepted, you need to do the following things:

Change the array nums such that the first k elements of nums
contain the elements which are not equal to val. The remaining elements
of nums are not important as well as the size of nums.


Example 1:
Input: nums = [3,2,2,3], val = 3
Output: 2, nums = [2,2,_,_]
Explanation: Your function should return k = 2,
with the first two elements of $nums$ being $2$.
It does not matter what you leave beyond the
returned k (hence they are underscores).

Example 2:
Input: nums = [0,1,2,2,3,0,4,2], val = 2
Output: 5, nums = [0,1,4,0,3,_,_,_]
Explanation: Your function should return k = 5,
with the first five elements of nums containing 0, 0, 1, 3, and 4.
Note that the five elements can be returned in any order.
It does not matter what you leave beyond
the returned k (hence they are underscores).
```



2. (10 points) "Non-decreasing array". Provide your explanation for a solution to the following problem. You do not need to give me the code.

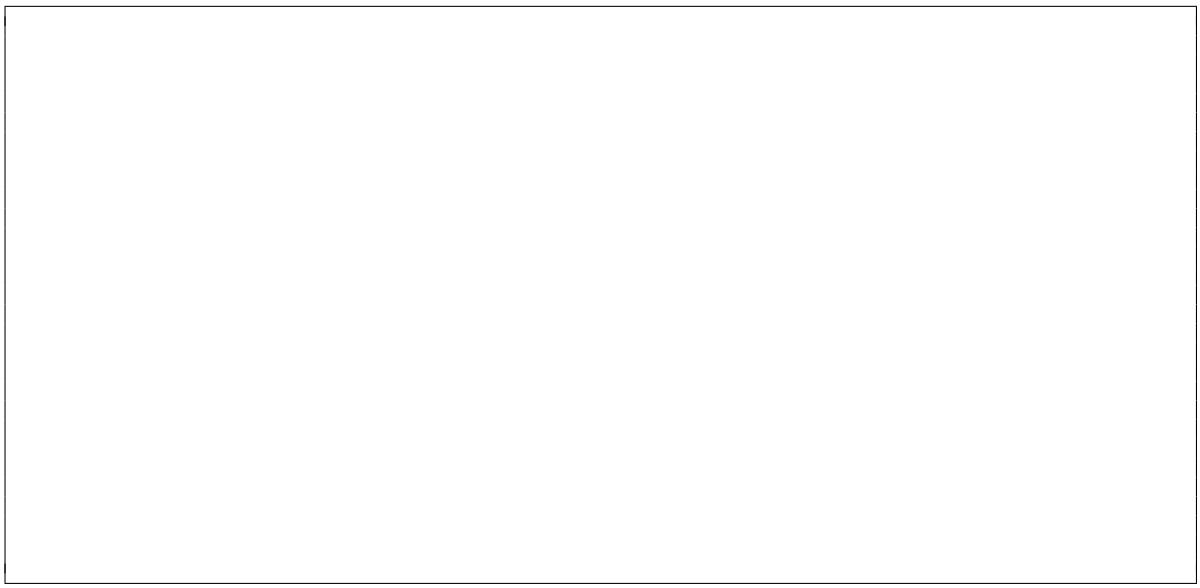
```
Given an array nums with n integers, your task is to check if it could become
non-decreasing by modifying at most one element.

We define an array is non-decreasing if nums[i] <= nums[i + 1] holds
for every i (0-based) such that (0 <= i <= n - 2).



Example 1:
Input: nums = [4,2,3]
Output: true
Explanation: You could modify the first 4 to 1 to get a
non-decreasing array.

Example 2:

Input: nums = [4,2,1]
Output: false
Explanation: You cannot get a non-decreasing array by modifying at most
one element.


Constraints:

n == nums.length
```

3. (10 points) "Climbing Stairs". Provide the code for the following question.

```
You are climbing a staircase. It takes n steps to reach the top.
Each time you can either climb 1 or 2 steps.
In how many distinct ways can you climb to the top?


Example 1:
Input: n = 2
Output: 2
Explanation: There are two ways to climb to the top.
1. 1 step + 1 step
2. 2 steps

Example 2:
Input: n = 3
Output: 3
Explanation: There are three ways to climb to the top.
1. 1 step + 1 step + 1 step
2. 1 step + 2 steps
3. 2 steps + 1 step

Constraints:

1 <= n <= 45
```

4. (10 points) "Implement Stack Using Queues". Provide your explanation for a solution to the following problem. You do not need to give me the code.

```
Implement a last-in-first-out (LIFO) stack using only two queues.
The implemented stack should support all the functions of a
normal stack (push, top, pop, and empty).

Implement the MyStack class:

void push(int x) Pushes element x to the top of the stack.
int pop() Removes the element on the top of the stack and returns it.
int top() Returns the element on the top of the stack.
boolean empty() Returns true if the stack is empty, false otherwise.
Notes:

You must use only standard operations of a queue, which means
that only push to back, peek/pop from front, size and is empty
operations are valid.
Depending on your language, the queue may not be supported natively.
You may simulate a queue using a list or deque (double-ended queue)
as long as you use only a queue's standard operations.


Example 1:

Input
["MyStack", "push", "push", "top", "pop", "empty"]
[[], [1], [2], [], [], []]
Output
[null, null, null, 2, 2, false]

Explanation
MyStack myStack = new MyStack();
myStack.push(1);
myStack.push(2);
myStack.top(); // return 2
myStack.pop(); // return 2
myStack.empty(); // return False
```

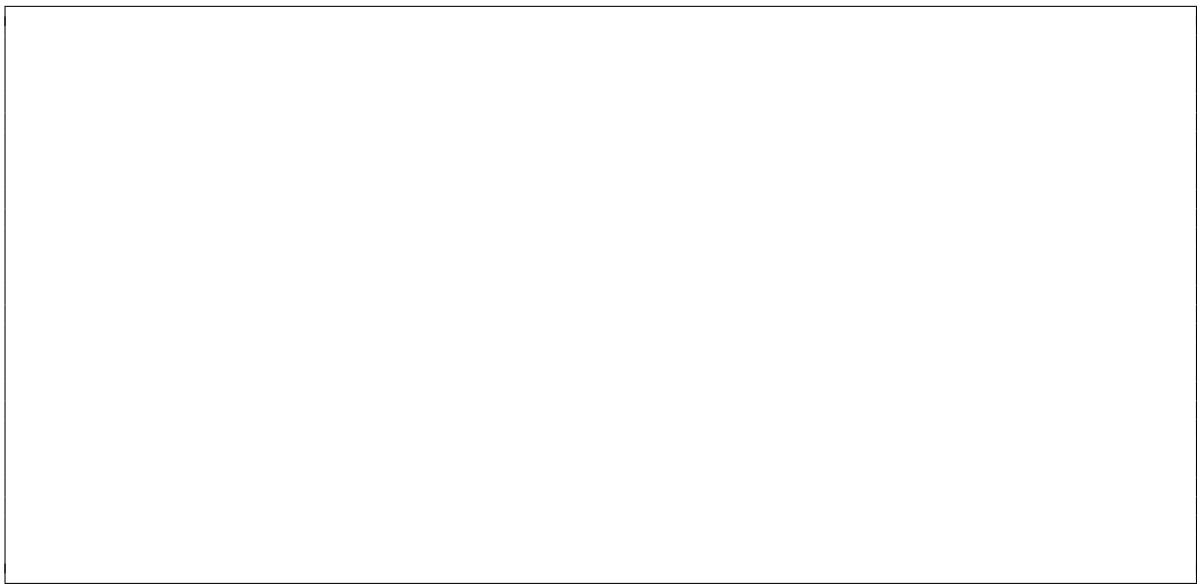


5. (10 points) "Sort an Array". Provide the code for the following question.

```
Given an array of integers nums, sort the array in ascending order and return it.
You must solve the problem without using any built-in functions in O(nlog(n))
time complexity and with the smallest space complexity possible.

Example 1:
Input: nums = [5,2,3,1]
Output: [1,2,3,5]
Explanation: After sorting the array, the positions
of some numbers are not changed (for example, 2 and 3),
while the positions of other numbers are changed (for example, 1 and 5).

Example 2:
Input: nums = [5,1,1,2,0,0]
Output: [0,0,1,1,2,5]
Explanation: Note that the values of nums are not necessairly unique.
```

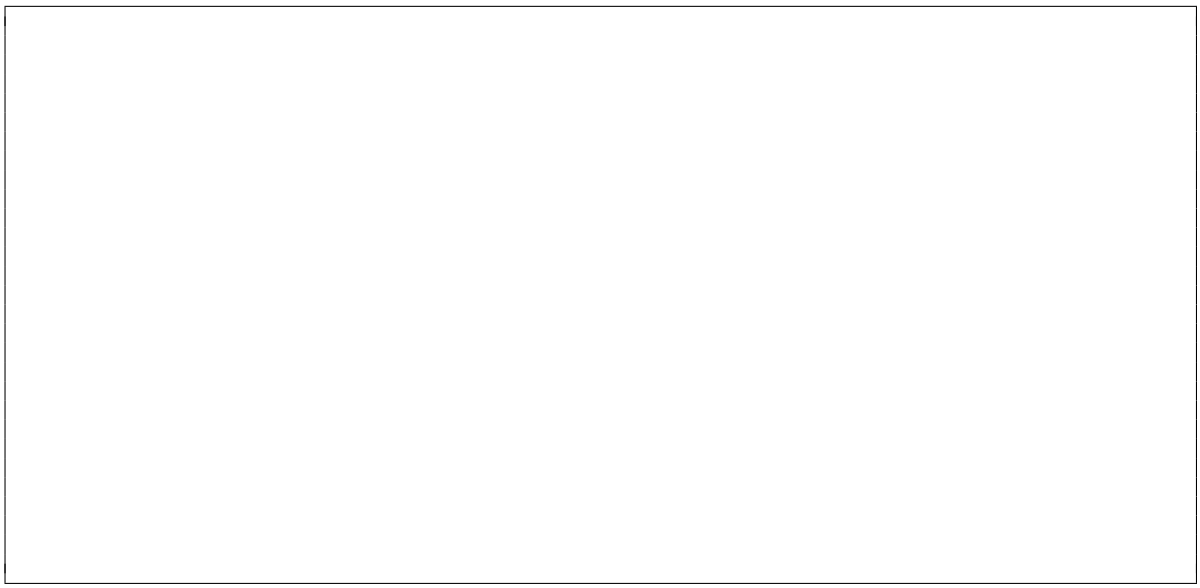
6. (10 points) "-Merge k- Sorted Lists". Provide your explanation for a solution to the following problem. You do not need to give me the code.

```
You are given an array of k linked-lists lists,
each linked-list is sorted in ascending order.

Merge all the linked-lists into one sorted linked-list and return it.

Example 1:
Input: lists = [[1,4,5],[1,3,4],[2,6]]
Output: [1,1,2,3,4,4,5,6]
Explanation: The linked-lists are:
[
  1->4->5,
  1->3->4,
  2->6
]
merging them into one sorted list:
1->1->2->3->4->4->5->6


Example 2:
Input: lists = []
Output: []


Example 3:
Input: lists = [[]]
Output: []

Constraints:
k == lists.length
lists[i] is sorted in ascending order.
```



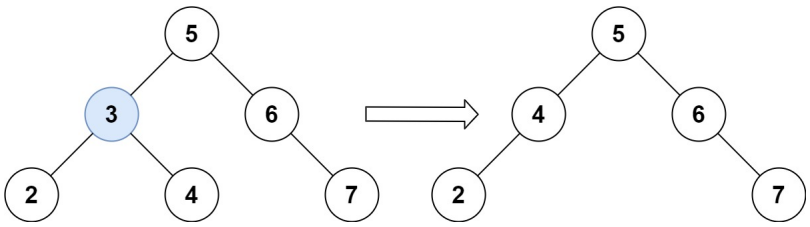
7. (10 points) "First Bad Version". Provide the code for the following question.

```
You are a product manager and currently leading a team
to develop a new product. Unfortunately, the latest version
of your product fails the quality check. Since each version
is developed based on the previous version, all the versions
after a bad version are also bad.

Suppose you have n versions [1, 2, ..., n] and you want to find
out the first bad one, which causes all the following ones to be bad.

You are given an API bool isBadVersion(version) which returns
whether version is bad. Implement a function to find the first
bad version. You should minimize the number of calls to the API.



Example 1:

- Input: n = 5, bad = 4
- Output: 4

Explanation:
- call isBadVersion(3) -> false
- call isBadVersion(5) -> true
- call isBadVersion(4) -> true
- Then 4 is the first bad version.

Example 2:

- Input: n = 1, bad = 1
- Output: 1
```

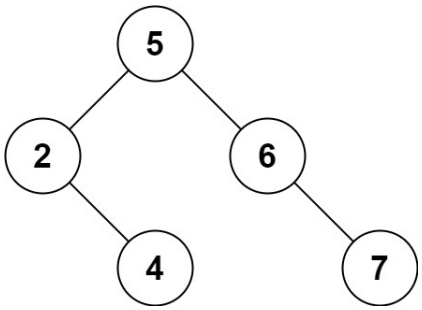


8. (10 points) "Delete Node in a BST". Provide your explanation for a solution to the following problem. You do not need to give me the code.

```
Given a root node reference of a BST and a key,
delete the node with the given key in the BST. Return
the root node reference (possibly updated) of the BST.
Basically, the deletion can be divided into two stages:

- Search for a node to remove.
- If the node is found, delete the node.

Example 1:
```




```
- Input: root = [5,3,6,2,4,null,7], key = 3
- Output: [5,4,6,2,null,null,7]

Explanation: Given key to delete is 3. So we find the node
with value 3 and delete it.
One valid answer is [5,4,6,2,null,null,7], shown in the above BST.
Please notice that another valid answer is [5,2,6,null,4,null,7]
and it's also accepted.

Example 2:
```




```
- Input: root = [5,3,6,2,4,null,7], key = 0
- Output: [5,3,6,2,4,null,7]
```

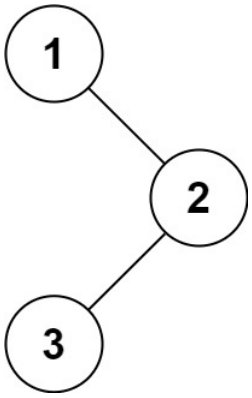
Explanation: The tree does not contain a node with value = 0.

Example 3:

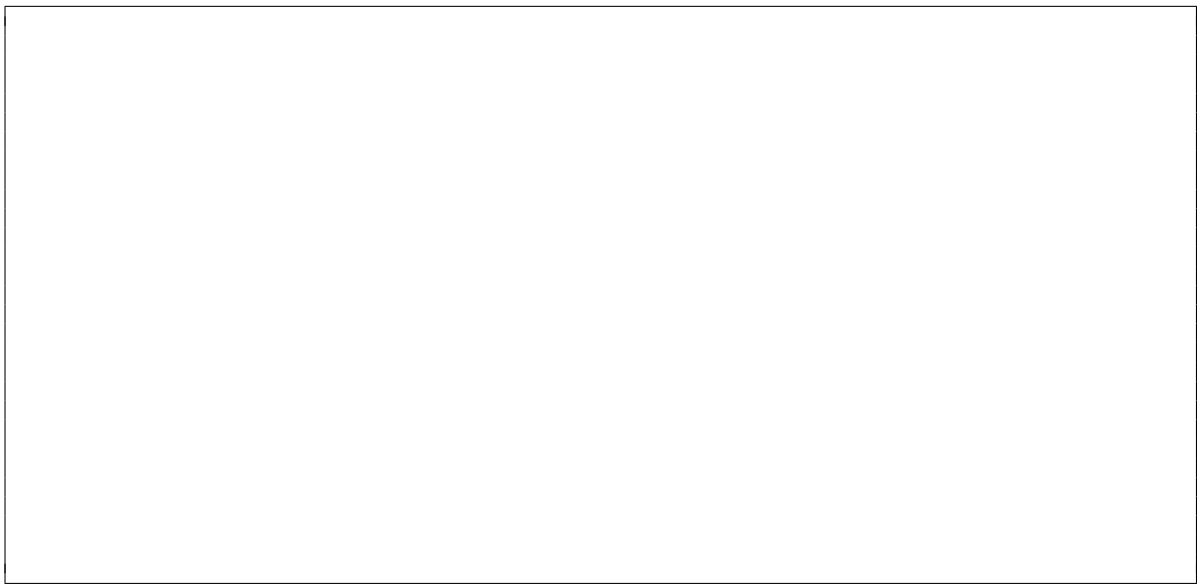
- Input: root = [], key = 0
- Output: []

9. (10 points) "Binary Tree Inorder Traversal". Provide the code for the following question.

```
Given the root of a binary tree, return the inorder traversal of its nodes' values.
```

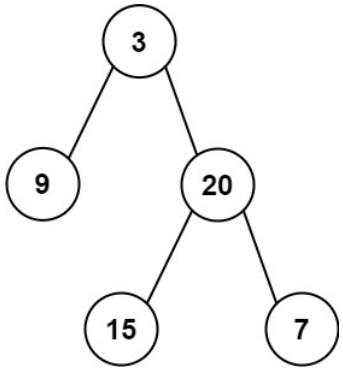



```
**Example 1:**

- Input: root = [1,null,2,3]
- Output: [1,3,2]

**Example 2:**

- Input: root = []
- Output: []

**Example 3:**

- Input: root = [1]
- Output: [1]


Constraints:

- The number of nodes in the tree is in the range [0, 100].
- -100 <= Node.val <= 100
```



10. (10 points) "Binary Tree Level Order Traversal". Provide your explanation for a solution to the following problem. You do not need to give me the code.

```
Given the root of a binary tree, return the level order traversal of
its nodes' values. (i.e., from left to right, level by level).

**Example 1:**
```




```
- Input: root = [3,9,20,null,null,15,7]
- Output: [[3],[9,20],[15,7]]

**Example 2:**

- Input: root = [1]
- Output: [[1]]

**Example 3:**

- Input: root = []
- Output: []
```

11. (10 points) "Contains Duplicate". Given an integer array nums, return true if any value appears at least twice in the array, and return false if every element is distinct.

```
- Example 1:
    - Input: nums = [1,2,3,1]
    - Output: true

- Example 2:
    - Input: nums = [1,2,3,4]
    - Output: false
- Example 3:
    - Input: nums = [1,1,1,3,3,4,3,2,4,2]
    - Output: true
```

12. (10 points) "LRU Cache-Medium". Design a data structure that follows the constraints of a Least Recently Used (LRU) cache.

```
Implement the LRUCache class:

LRUCache(int capacity) Initialize the LRU cache with positive size capacity.
int get(int key) Return the value of the key if the key exists, otherwise return -1.
void put(int key, int value) Update the value of the key if the key exists.
Otherwise,add the key-value pair to the cache. If the number of keys exceeds
the capacity from this operation, evict the least recently used key.

The functions get and put must each run in O(1) average time complexity.



- Example 1:

    - Input ["LRUCache", "put", "put", "get", "put",
    "get", "put", "get", "get", "get"] [[2], [1, 1], [2, 2], [1],
    [3, 3], [2], [4, 4], [1], [3], [4]]
    - Output
    [null, null, null, 1, null, -1, null, -1, 3, 4]

    - Explanation
        - LRUCache lRUCache = new LRUCache(2);
        - lRUCache.put(1, 1); // cache is {1=1}
        - lRUCache.put(2, 2); // cache is {1=1, 2=2}
        - lRUCache.get(1);    // return 1
```

```
- lRUCache.put(3, 3); // LRU key was 2, evicts key 2, cache is {1=1, 3=3}
- lRUCache.get(2);    // returns -1 (not found)
- lRUCache.put(4, 4); // LRU key was 1, evicts key 1, cache is {4=4, 3=3}
- lRUCache.get(1);    // return -1 (not found)
- lRUCache.get(3);    // return 3
- lRUCache.get(4);    // return 4
```

13. (10 points) "Kth Largest Element in a Stream". Design a class to find the kth largest element in a stream. Note that it is the kth largest element in the sorted order, not the kth distinct element.

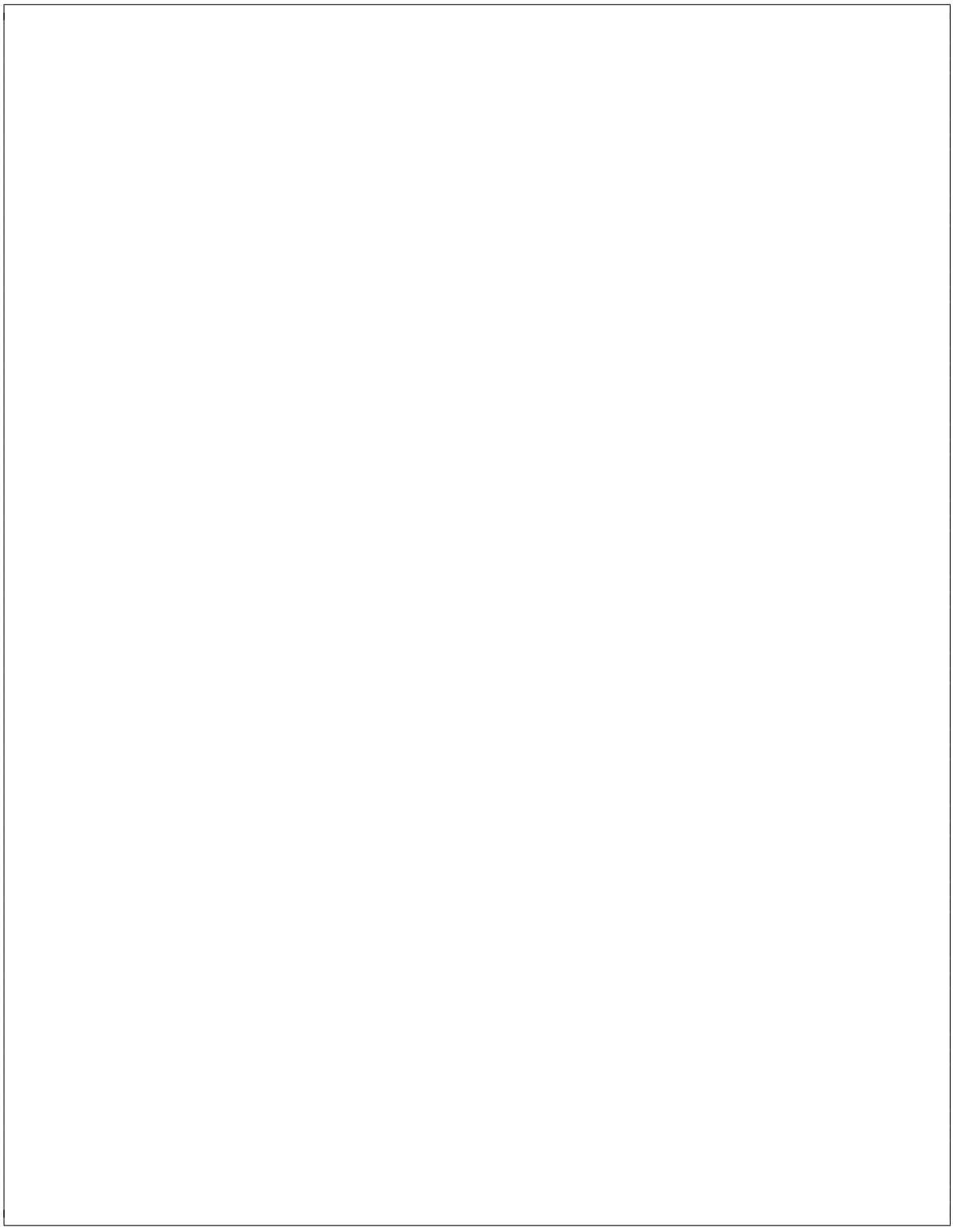
```
Implement KthLargest class:

KthLargest(int k, int[] nums) Initializes the object with the integer k and the strea
int add(int val) Appends the integer val to the stream and returns the element repres


**Example 1:**

- Input
    - ["KthLargest", "add", "add", "add", "add", "add"]
    - [[3, [4, 5, 8, 2]], [3], [5], [10], [9], [4]]
- Output
    - [null, 4, 5, 5, 8, 8]

- Explanation
    -KthLargest kthLargest = new KthLargest(3, [4, 5, 8, 2]);
    -kthLargest.add(3);   // return 4
    -kthLargest.add(5);   // return 5
    -kthLargest.add(10);  // return 5
    -kthLargest.add(9);   // return 8
    -kthLargest.add(4);   // return 8


- Constraints:

    - 1 <= k <= 104
    - 0 <= nums.length <= 104
    - -104 <= nums[i] <= 104
    - -104 <= val <= 104
    - At most 104 calls will be made to add.
    - It is guaranteed that there will be at least k elements in the array when you s
```

14. (10 points) "Number of Islands". Given an m x n 2D binary grid grid which represents a map of '1's (land) and '0's (water), return the number of islands.

    An island is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

```
- Example 1:

Input: grid = [\
  ["1","1","1","1","0"],\
  ["1","1","0","1","0"],\
  ["1","1","0","0","0"],\
  ["0","0","0","0","0"]\
]

Output: 1
- Example 2:

Input: grid = [\
  ["1","1","0","0","0"],\
  ["1","1","0","0","0"],\
  ["0","0","1","0","0"],\
  ["0","0","0","1","1"]\
]

Output: 3


- Constraints:
```

- `m == grid.length`
- `n == grid[i].length`



15. (10 points) "Course Schedule". There are a total of numCourses courses you have to take, labeled from 0 to numCourses - 1. You are given an array prerequisites where prerequisites[i] = [ai, bi] indicates that you must take course bi first if you want to take

course ai.

For example, the pair [0, 1], indicates that to take course 0 you have to first take course 1. Return true if you can finish all courses. Otherwise, return false.

- Example 1:

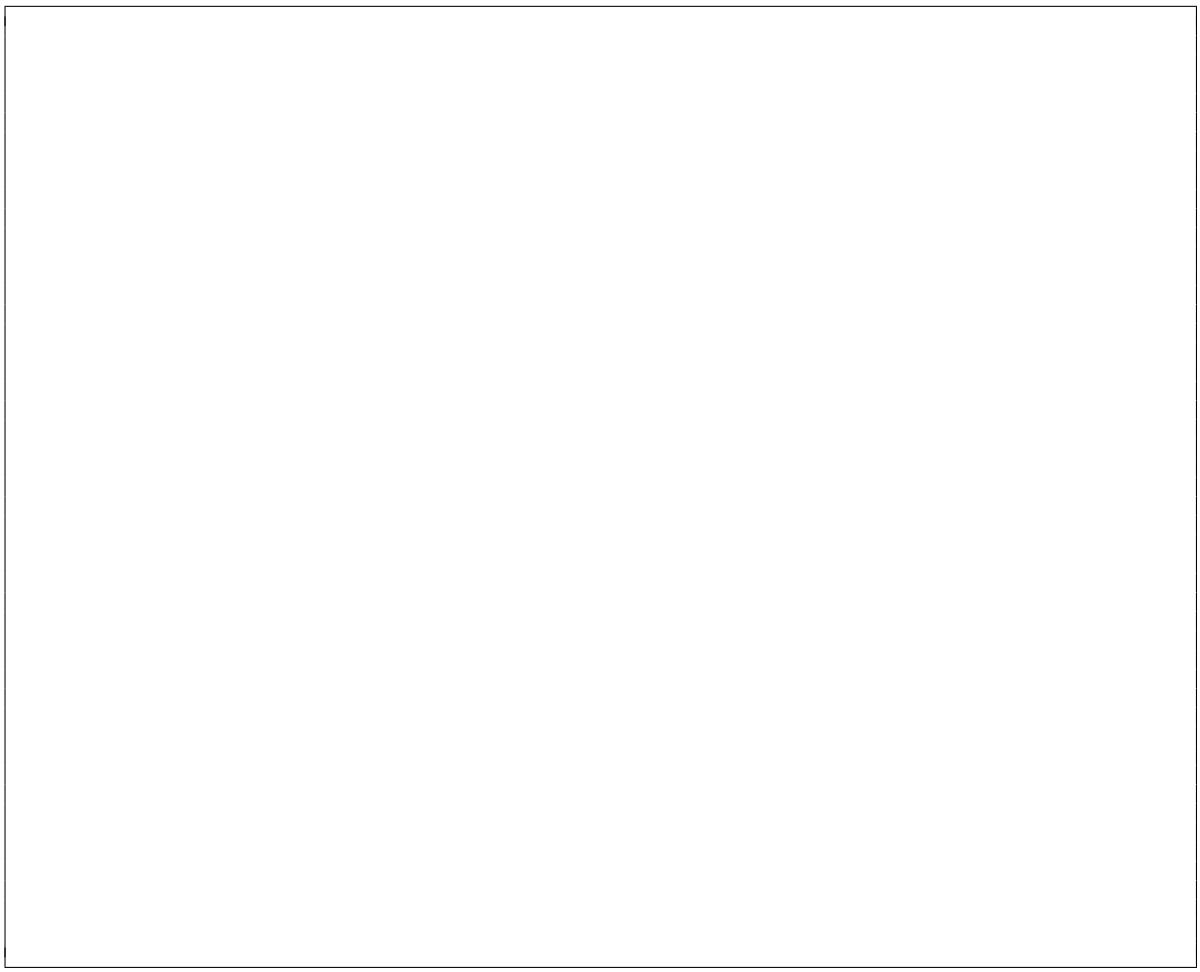
    - Input: numCourses = 2, prerequisites = [[1,0]]
    - Output: true
    - Explanation: There are a total of 2 courses to take. \
      To take course 1 you should have finished course 0. So it is possible.

- Example 2:

    - Input: numCourses = 2, prerequisites = [[1,0],[0,1]]
    - Output: false
    - Explanation: There are a total of 2 courses to take. \
      To take course 1 you should have finished course 0,\
      and to take course 0 you should also have finished course 1. So it is impossibl

- Constraints:

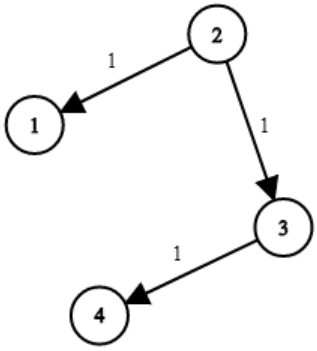
    - 1 <= numCourses <= 2000
    - 0 <= prerequisites.length <= 5000
    - prerequisites[i].length == 2
    - 0 <= ai, bi < numCourses
    - All the pairs prerequisites[i] are unique.



16. (10 points) "Network Delay Time". You are given a network of n nodes, labeled from 1 to n. You are also given times, a list of travel times as directed edges times[i] = (ui, vi, wi), where ui is the source node, vi is the target node, and wi is the time it takes for a signal to travel from source to target.

    We will send a signal from a given node k. Return the minimum time it takes for all the n nodes to receive the signal. If it is impossible for all the n nodes to receive the signal, return -1.

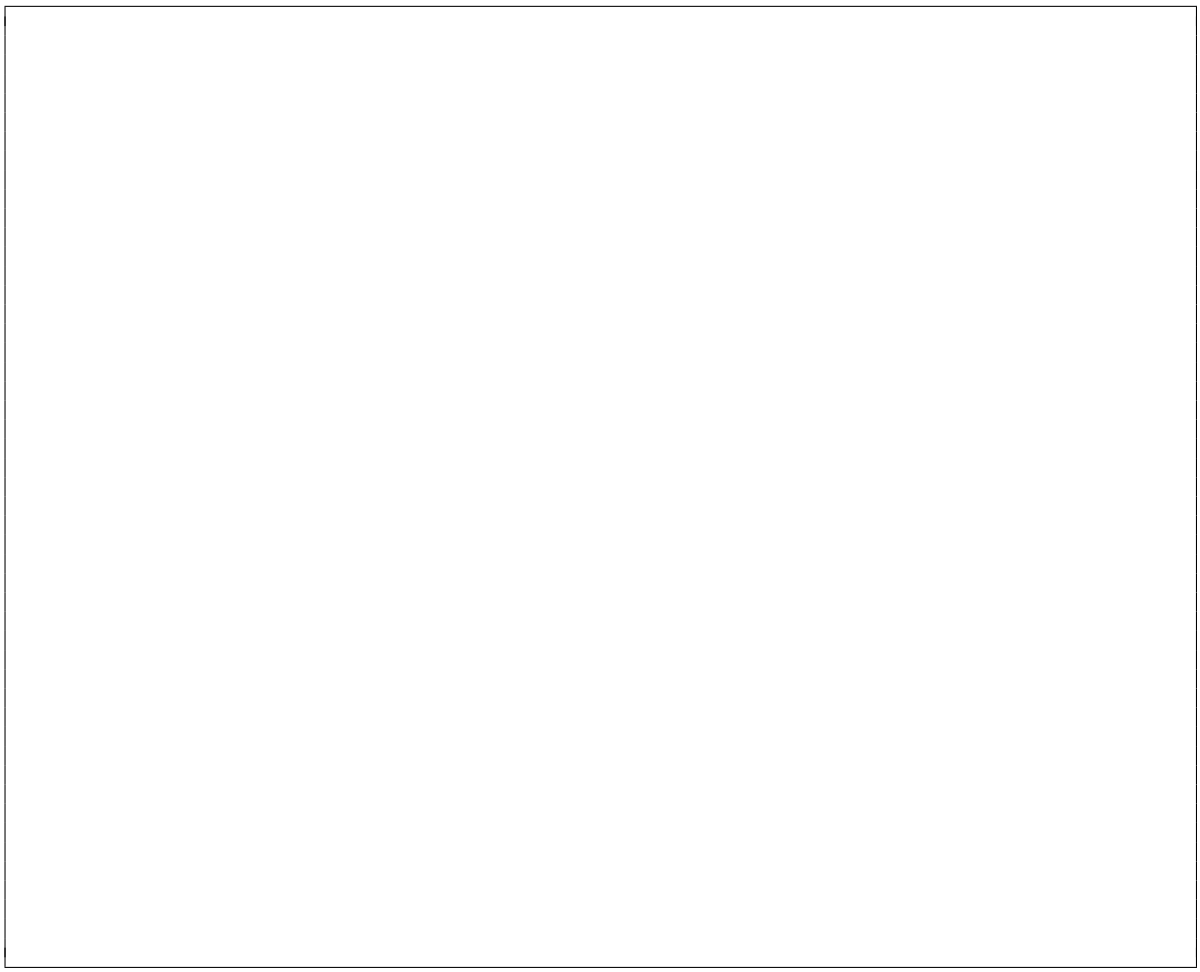
    - Example 1:



    - Input: times = [[2,1,1],[2,3,1],[3,4,1]], n = 4, k = 2
    - Output: 2
- Example 2:

    - Input: times = [[1,2,1]], n = 2, k = 1
    - Output: 1
- Example 3:

    - Input: times = [[1,2,1]], n = 2, k = 2
    - Output: -1



17. (10 points) "Course Schedule". There are a total of numCourses courses you have to take, labeled from 0 to numCourses - 1. You are given an array prerequisites where prerequisites[i] = [ai, bi] indicates that you must take course bi first if you want to take course ai.

    For example, the pair [0, 1], indicates that to take course 0 you have to first take course 1. Return true if you can finish all courses. Otherwise, return false.

```
 - Example 1:

    - Input: numCourses = 2, prerequisites = [[1,0]]
    - Output: true
    - Explanation: There are a total of 2 courses to take.
      To take course 1 you should have finished course 0. So it is possible.

- Example 2:

    - Input: numCourses = 2, prerequisites = [[1,0],[0,1]]
    - Output: false
    - Explanation: There are a total of 2 courses to take.
      To take course 1 you should have finished course 0,\
      and to take course 0 you should also have finished course 1.\
      So it is impossible.
```

18. (10 points) "Course Schedule II". There are a total of numCourses courses you have to take, labeled from 0 to numCourses - 1. You are given an array prerequisites where prerequisites[i] = [ai, bi] indicates that you must take course bi first if you want to take course ai.

    For example, the pair [0, 1], indicates that to take course 0 you have to first take course 1. Return the ordering of courses you should take to finish all courses. If there are many valid answers, return any of them. If it is impossible to finish all courses, return an empty array.

```
- Example 1:

    - Input: numCourses = 2, prerequisites = [[1,0]]
    - Output: [0,1]
    - Explanation: There are a total of 2 courses to take.\
        To take course 1 you should have finished course 0.
        So the correct course order is [0,1].

- Example 2:

    - Input: numCourses = 4, prerequisites = [[1,0],[2,0],[3,1],[3,2]]
    - Output: [0,2,1,3]
    - Explanation: There are a total of 4 courses to take. To take course 3\
        you should have finished both courses 1 and 2. Both courses 1 and 2\
        should be taken after you finished course 0.
        So one correct course order is [0,1,2,3].
        Another correct ordering is [0,2,1,3].

- Example 3:

    - Input: numCourses = 1, prerequisites = []
    - Output: [0]
```