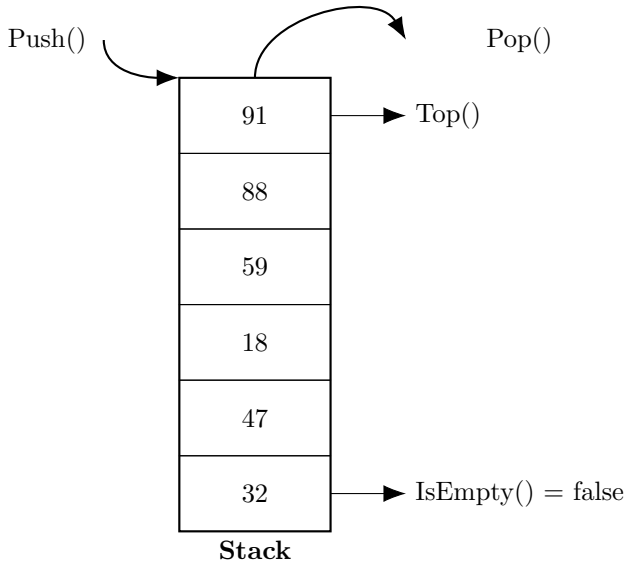
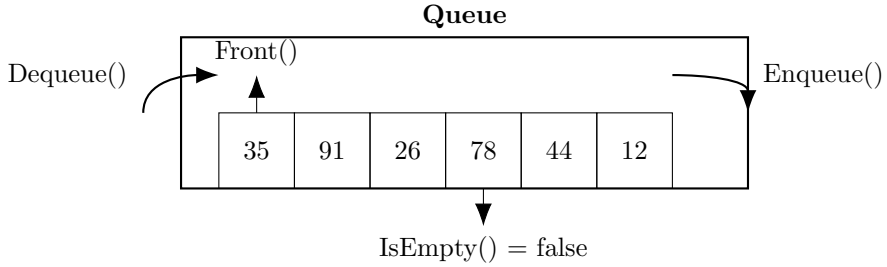
# Lab 2: Stack-Queue

## 1 What is Stack?Queue?

**The stack** is a Last In First Out **(LIFO)** data type.




**The Queue** data type is also known as the First In First Out (FIFO) data type since the element that is inserted from the back of Queue will travel to the front side before it can be removed.




## 2 In-Class Exercise - Part 1

Write a program to implement a stack using a singly linked list (or an array) to temporarily store integers (or characters). Perform the following requirements:

1. Write a function to initialize the stack.
2. Write a function to check if the stack is empty. If empty, return true, otherwise return false.
3. Write a function to check if the stack is full (if applicable). If full, return true, otherwise return false.
4. Write a function to push an element onto the stack.
5. Write a function to pop an element from the stack.

PREPARED BY Tran Vinh Khiem

6. Print the TOP element of the stack.

In the main function, write a menu to display the above choices.

# 3 In-Class Exercise - Part 2

Write a program to implement a queue using a singly linked list (or an array) to temporarily store integers (or characters). Perform the following requirements:

1. Write a function to initialize the queue.
2. Write a function to check if the queue is empty. If empty, return true, otherwise return false.
3. Write a function to check if the queue is full (if applicable). If full, return true, otherwise return false.
4. Write a function to enqueue an element into the queue.
5. Write a function to dequeue an element from the queue.
6. Print the REAR and FRONT elements of the queue.
7. In the main function, write a menu to display the above choices.

# 4 Homework

## 4.1 Question 1: Tower of Hanoi

In the classic problem of the Towers of Hanoi, you have 3 towers and N disks of different sizes which can slide onto any tower. The puzzle starts with disks sorted in ascending order of size from top to bottom (i.e., each disk sits on top of an even larger one).

**Constraints:**

Illustration:

- `https://upload.wikimedia.org/wikipedia/commons/4/4f/Tower_of_Hanoi.gif`
- `https://upload.wikimedia.org/wikipedia/commons/6/60/Tower_of_Hanoi_4.gif`

## Input

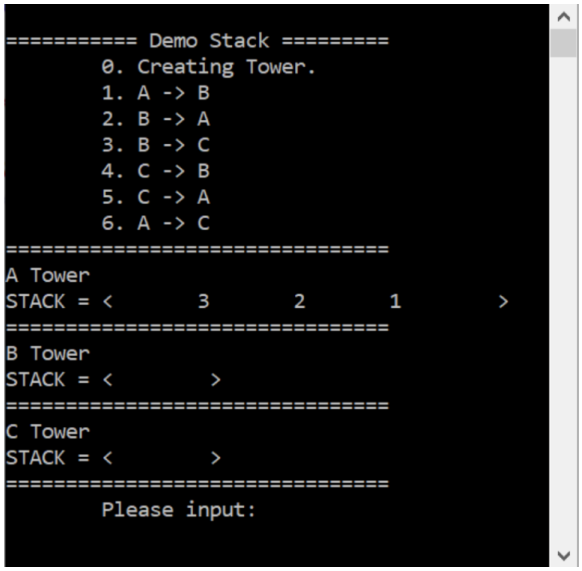



Figure 1: Initial status

## Output

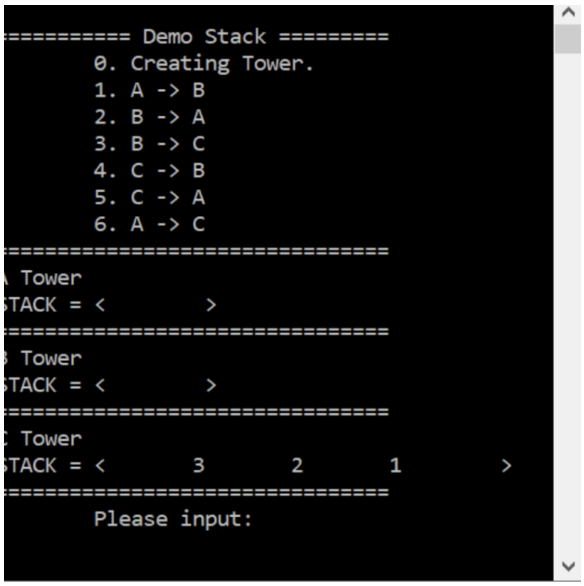



Figure 2: Finished status

## Constraints

(1) Only one disk can be moved at a time.
(2) A disk is slid off the top of one tower onto another tower.

PREPARED BY Tran Vinh Khiem

(3) A disk cannot be placed on top of a smaller disk. Write a program to move the disks from the first tower to the last using Stacks.

## 4.2 Question 2: Browser History Management

**Problem Statement:**

Simulate a browser's back and forward navigation using a stack. Implement the following functionalities:

- visit(url): Visit a new URL.
- back(): Go back to the previous URL.
- forward(): Go forward to the next URL.

**Example:**

```
browser.visit("httt.uit.edu.vn")
browser.visit("uit.edu.vn")
browser.back() // returns "httt.uit.edu.vn"
browser.forward() // returns "uit.edu.vn
browser.visit("daa.uit.edu.vn")
browser.back() // returns "uit.edu.vn"
```

## 4.3 Question 3

The problem involves a queue of $n$ people, each with a distinct height, arranged from left to right. Given an array called `heights`, where `heights[i]` represents the height of the $i$-th person, the task is to determine how many people each person can see to their right. A person can see another if everyone in between is shorter than both of them. Specifically, the $i$-th person can see the $j$-th person if $i < j$ and all heights between them are less than both `heights[i]` and `heights[j]`.

The solution should return an array `answer`, where `answer[i]` indicates the number of people visible to the $i$-th person on their right.

### Example 1:

```
Input: heights = [10,6,8,5,11,9]
Output: [3,1,2,1,0]
Explanation:
Person 0 can see person 1, 2, and 4.
Person 1 can see person 2.
Person 2 can see person 3 and 4.
Person 3 can see person 4.
Person 4 can see person 5.
Person 5 can see no one since nobody is to the right of them.
```

### Example 2:

```
Input: heights = [5,1,2,3,10]
Output: [4,1,1,1,0]
```

### Constraints:

- $n =$ `heights.length`
- $1 \leq n \leq 10^5$
- $1 \leq$ `heights`$[i] \leq 10^5$
- All values of `heights` are unique.

## 4.4 Question 4

You are provided with an integer array `prices` where `prices[i]` represents the number of coins required to buy the *i-th* fruit.

The fruit market offers the following incentive for each fruit:

If you buy the *i-th* fruit for `prices[i]` coins, you can obtain any number of the subsequent $(i + 1)$ fruits at no cost.

It is important to note that even if you can acquire fruit $j$ for free, you still have the option to purchase it for `prices[j]` coins to gain its reward.

Your task is to determine the minimum number of coins necessary to obtain all the fruits.

**Example 1:**

Input: `prices = [3,1,2]`

Output: 4

Explanation:

Buy the 1st fruit for `prices[0]` = 3 coins, allowing you to take the 2nd fruit for free. Buy the 2nd fruit for `prices[1]` = 1 coin, permitting you to take the 3rd fruit for free. Take the 3rd fruit at no cost. Note that although you could have taken the 2nd fruit for free as a reward for purchasing the 1st fruit, buying it is more advantageous for receiving its reward.

**Example 2:**

Input: `prices = [1,10,1,1]`

Output: 2

Explanation:

Buy the 1st fruit for `prices[0]` = 1 coin, allowing you to take the 2nd fruit for free. Take the 2nd fruit at no cost. Buy the 3rd fruit for `prices[2]` = 1 coin, permitting you to take the 4th fruit for free. Take the 4th fruit at no cost.

**Example 3:**

Input: `prices = [26,18,6,12,49,7,45,45]`

Output: 39

Explanation:

Buy the 1st fruit for `prices[0]` = 26 coins, allowing you to take the 2nd fruit for free. Take the 2nd fruit at no cost. Buy the 3rd fruit for `prices[2]` = 6 coins, permitting you to take the next three fruits (4th, 5th, and 6th) at no cost. Take the 4th fruit at no cost. Take the 5th fruit at no cost. Buy the 6th fruit for `prices[5]` = 7 coins, allowing you to take both the 7th and 8th fruits at no cost. Take the 7th and 8th fruits at no cost. Note that although you could have taken the 6th fruit for free as a reward from buying the 3rd fruit, purchasing it is more optimal to receive its reward.

**Constraints:**

- $1 \leq$ `prices.length` $\leq 1000$
- $1 \leq$ `prices[i]` $\leq 10^5$

## Notice

- Use C++ for practice.
- In the programming file, the student should include the following complete information:

```
//STT: 39 (Example)
//Full Name: X, With X is you, don't need to find X anywhere else.
//Session 01 - Exercise 01
//Notes or Remarks: ......
```

## References :

[1]. Skiena, S. S. (1998). The algorithm design manual (Vol. 2). New York: springer.
[2]. Pai, G. V. (2023). A Textbook of Data Structures and Algorithms, Volume 3: Mastering Advanced Data Structures and Algorithm Design Strategies. John Wiley & Sons.
[3]. Anggoro, W. (2018). C++ Data Structures and Algorithms: Learn how to write efficient code to build scalable and robust applications in C++. Packt Publishing Ltd.
[3].Leetcode
[4].Codeforce