DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

# Experiment-1

**Student Name: Anupreet Kaur** **UID: 22BCS50071**

**Branch: BE-CSE** **Section/Group: 22BCS_NTPP_602-A**

**Semester: 6th** **Date of Performance: 20/01/2025**

**Subject Name: AP Lab** **Subject Code: 22CSP-351**

## 1. Aim: Arrays, Stacks, Queues.

- ❖ Problem 1.2.1: Two Sum- Given an array of integers nums and an integer target, return the indices of the two numbers such that they add up to target. Each input has exactly one solution, and you cannot use the same element twice.
- ❖ Problem 1.2.2: Implement Queue Using Stacks- Implement a FIFO queue using two stacks. The queue should support all the functions of a normal queue: push, peek, pop, and empty.

## 2. Objective:

To understand and implement fundamental data structures such as arrays, stacks, and queues to solve real-world computational problems efficiently. Specifically, solving problems like finding pairs of numbers that sum up to a target value and implementing a queue using stacks to demonstrate the versatility of data structures.

## 3. Theory:

**Arrays:** Arrays are linear data structures that store elements of the same data type in contiguous memory locations.

- Used for storing and accessing data efficiently by index.
- Key operations include traversal, searching, insertion, and deletion.

**Stacks:** A stack is a Last-In-First-Out (LIFO) data structure where elements are added (pushed) and removed (popped) from the top.

- Common operations: push, pop, peek, isEmpty.
- Applications include function call management, expression evaluation, and reversing strings.

**Queues:** A queue is a First-In-First-Out (FIFO) data structure where elements are added at the rear and removed from the front.

- Common operations: enqueue, dequeue, peek, isEmpty.
- Applications include job scheduling, buffering in communication systems, and level-order traversal of trees.

## 4. Code:

### Two Sum

```
class Solution {
    public int[] twoSum(int[] nums, int target) {
        for (int i = 0; i < nums.length; i++) {
            for (int j = i + 1; j < nums.length; j++) {
                if (nums[i] + nums[j] == target) {
                    return new int[] { i, j };
                }
            }
        }
        throw new IllegalArgumentException("No two sum solution");
    }
    }
```

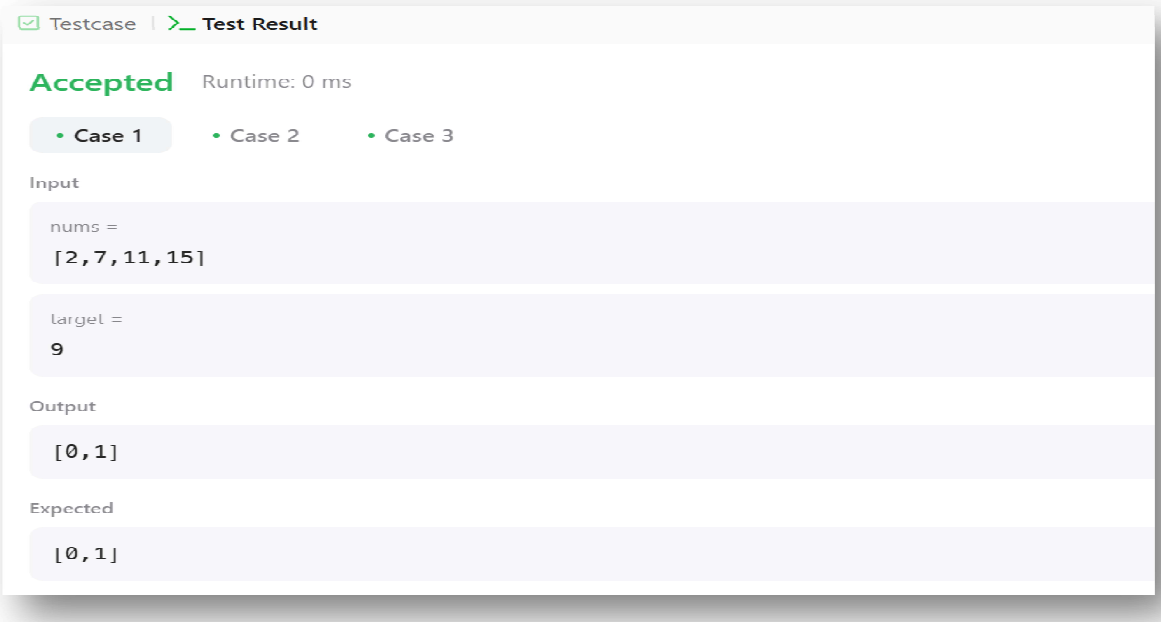
### Implement Queue using Stacks

```
import java.util.Stack;

class MyQueue {
    private Stack<Integer> Stack1;
    private Stack<Integer> Stack2;
    public MyQueue() {
        Stack1 = new Stack<>();
        Stack2 = new Stack<>();
    }
    public void push(int x) {
        Stack1.push(x);
    }
    public int pop() {
        shiftStacks();
        return Stack2.pop();
    }
    public int peek() {
        shiftStacks();
        return Stack2.peek();
    }

    // Returns true if the queue is empty, false otherwise
    public boolean empty() {
        return Stack1.isEmpty() && Stack2.isEmpty();
    }
    private void shiftStacks() {
        if (Stack2.isEmpty()) {
            while (!Stack1.isEmpty()) {
                Stack2.push(Stack1.pop());
            }
        }
    }
    }
```

## 6. Output:





## 7. Learning Outcomes:

- Understand the differences and use cases of arrays, stacks, and queues.
- Solve the Two Sum problem to enhance understanding of array manipulation.
- Implement a queue using stacks to explore the interrelation between data structures.
- Recognize how these structures optimize data management and access.
- Develop efficient algorithms for the given problems with constraints like time complexity and memory optimization.