Experiment 2

Student Name: Ankita **UID:** 22BCS17162
Branch: CSE **Section/Group:**IOT_NTPP_602-A
Semester: 6th **Date of Performance:**20-01-25
Subject Name: AP- 2 **Subject Code:** 22CSP-351

Aim: a) 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.

b) Implement a FIFO queue using two stacks. The queue should support all the functions of a normal queue: push, peek, pop, and empty.

Objective: a) Develop an efficient algorithm to find two indices in an integer array whose elements sum to a given target, ensuring optimal performance and avoiding repeated use of same element.

b) Implement a First-In-First-Out (FIFO) queue using two stacks, ensuring support for essential operations—push, peek, pop, and empty—while maintaining correct order and efficient execution.

Algorithm:

a) Two sum problem

1. Initialize an empty hashmap to store numbers and their indices.

2. Iterate through the array:

- Calculate the complement by subtracting the current element from the target.
- Check if the complement exists in the hashmap.
- If found, return the indices of the complement and current element.
- Otherwise, store the current number with its index in the hashmap.

3. Return an empty array.

b) Implement Queue using Stacks

1.Initialize two stacks, stack1 for enqueue operations and stack2 for dequeue operations.
2.For push(x), add the element to stack1.

3.For pop(), if stack2 is empty, transfer all elements from stack1 to stack2, then remove the top of stack2.
4.For peek(), if stack2 is empty, transfer elements from stack1 to stack2, then return the top of stack2
5. For empty(), check if both stack1 and stack2 are empty.

Code:

a)

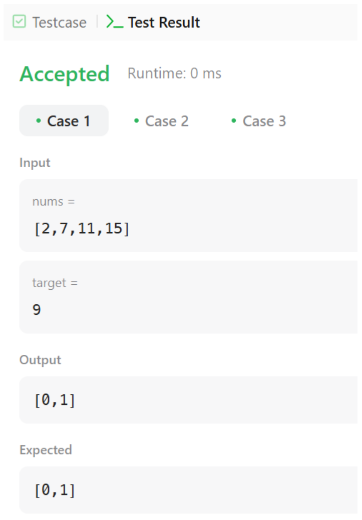
```
#include <vector>
#include <unordered_map>
class Solution {
public:
std::vector<int> twoSum(const std::vector<int>& nums, int target) {
std::unordered_map<int, int> numMap;
for (int i = 0; i < nums.size(); ++i) {
int complement = target - nums[i];
if (numMap.find(complement) != numMap.end()) {
return {numMap[complement], i};
}
numMap[nums[i]] = i;
}
return {};
}
};
```

b)

```
#include <stack>
using namespace std;
class MyQueue {
private:
stack<int> stack1;
stack<int> stack2;
void change()
{ while(!stack1.empty())
{ stack2.push(stack1.top());
stack1.pop();
}
}
public:
MyQueue()
{}
void push(int x) {
stack1.push(x);
}
int pop() {
if (stack2.empty()) {
change();
}
int front=stack2.top();
stack2.pop();
return front;
}
int peek() {
if (stack2.empty()) {
change();}
return stack2.top();
}
bool empty() {
return stack1.empty() && stack2.empty();
}
};
```

Output:

a)



Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

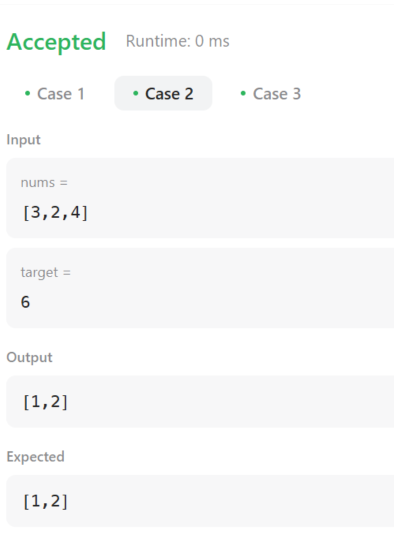
```
nums =
[2,7,11,15]
target =
9
```

Output

```
[0,1]
```

Expected

```
[0,1]
```



Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

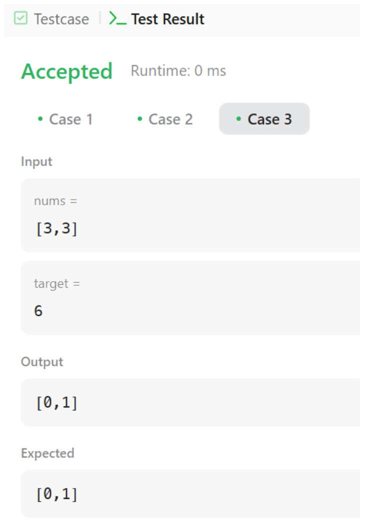
```
nums =
[3,2,4]
target =
6
```

Output

```
[1,2]
```

Expected

```
[1,2]
```



Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

```
nums =
[3,3]
target =
6
```

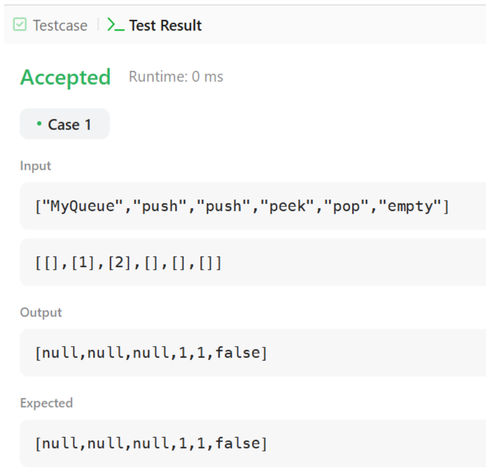
Output

```
[0,1]
```

Expected

```
[0,1]
```

b)



Accepted Runtime: 0 ms

Case 1

Input

```
["MyQueue","push","push","peek","pop","empty"]
[[],[1],[2],[],[],[]]
```

Output

```
[null,null,null,1,1,false]
```

Expected

```
[null,null,null,1,1,false]
```

Learning Outcomes:

a) Learn to efficiently find pairs of numbers in an array that sum to a target using a hashmap.
b) Understand the concept of complementing values to optimize search operations in arrays.
c) Gain proficiency in handling edge cases and constraints in algorithmic problems.
d) Master the implementation of a FIFO queue using two stacks, ensuring correct operation of queue functions.