Abstract:

This project presents solutions to LeetCode problems focusing on various data structures including arrays, strings, integers, stacks, queues, and linked lists. Each problem statement is accompanied by a C++ code implementation, which has been rigorously tested and accepted by LeetCode. The problems are designed to cover fundamental concepts in data structures and algorithms, providing a comprehensive learning experience for students in a Data Structures Laboratory course (ISL36). The solutions demonstrate efficient problem-solving techniques and adhere to the specified constraints and requirements.

Note on Language Choice:

C++ was chosen over C due to its robust standard library, which provides powerful data structures and algorithms implementations, making it more suitable for solving complex problems efficiently. Additionally, C++ offers features like object-oriented programming, which can help in organizing and structuring code in a more modular and maintainable way.

Libraries Used:

1. **`<vector>`**: Used for dynamic array manipulation and storage in the array and queue implementations.
2. **`<string>`**: Utilized for string processing and manipulation in the string shuffle problem.
3. **`<algorithm>`**: Employed for searching and manipulation algorithms like find and reverse in the string shuffle and number palindromic problems.
4. **`<deque>`**: Used for implementing double-ended queues in the queue problem for efficient insertion and deletion operations at both ends.

5. **<bits/stdc++.h>**: A convenience header that includes most standard C++ libraries, used for general-purpose coding convenience and compatibility across different compilers.

Code:

Problem 1: Arrays (Difficulty: Easy)

Given an array nums consisting of 2n elements in the form [x1,x2,...,xn,y1,y2,...,yn], return the array in the form [x1,y1,x2,y2,...,xn,yn].

Solution Code:

```
#include<bits/stdc++.h>
#include<vector>
using namespace std;

class Solution {
    public:
    vector<int> a;
    vector<int> shuffle(vector<int>& a, int n) {
    vector<int> f(a.begin(), a.begin()+(a.size()/2));
    vector<int> l(a.begin()+(a.size()/2), a.end());
    int j = 1;
    for (int i = 0; i < l.size(); i++)
    {
        f.insert(f.begin()+j, l[i]);
        j = j+2;
    }
    return f;
    }
};
```

```
};
```

Problem 2: Strings and Integers

Statement (i): Strings (Difficulty: Easy)

Given a string s and an integer array indices of the same length, shuffle the string such that the character at the ith position moves to indices[i] in the shuffled string.

Solution Code:

```
#include<bits/stdc++.h>
#include<vector>
#include<string>
#include<algorithm>
using namespace std;

class Solution {
public:
     string restoreString(string s, vector<int>& indices) {
     string res;
     for(int i = 0; i < s.length(); i++)
     {
          auto it = find(indices.begin(), indices.end(), i);
          int ind = distance(indices.begin(),it);
          res.push_back(s[ind]);
     }
     return res;
     }
};
```

Statement (ii): Integers (Difficulty: Medium)

Determine if an integer n is strictly palindromic, i.e., for every base b between 2 and n - 2, the string representation of n in base b is palindromic.

Solution Code:

```
#include<bits/stdc++.h>
#include<string>
#include<deque>
using namespace std;

class Solution {
public:
    bool isStrictlyPalindromic(int n) {
    deque<int> basen;
    for (int base = 2; base <= n -2; base++)
    {
        int t = n;
        int d;
        while(t>0)
        {
            d = t%base;
            basen.push_back(d);
            t = t/base;
        }
        deque<int> temp = basen;
        reverse(basen.begin(), basen.end());
        if (temp != basen)
        return false;
    }
    return true;
    }
};
```

Problem 3: Stacks (Difficulty: Medium)

Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

Solution Code:

```
#include<bits/stdc++.h>
#include<vector>
using namespace std;

class MinStack {
    public:
    vector<int> s;
    int min;

    MinStack() {
    min = 0;
    }

    void push (int val) {
    s.push_back(val);
    }

    void pop () {
    s.pop_back();
    }

    int top () {
    return s[s.size()-1];
    }

    int getMin() {
    min = s[0];
    for(int i = 0; i < s.size(); i++)
    {
        if(s[i] < min)
            min = s[i];
    }
    return min;
    }
};
```

Problem 4: Queues (Difficulty: Medium)

Design a queue that supports push and pop operations in the front, middle, and back.

Solution Code:

```
#include<bits/stdc++.h>
#include<deque>
using namespace std;

class FrontMiddleBackQueue {
public:
    deque<int> q;


    FrontMiddleBackQueue() {}

    void pushFront(int val) {
    q.push_front(val);
    }

    void pushMiddle(int val) {
    int mid = q.size()/2;
    auto middle = q.begin() + mid;
    q.insert(middle, val);
    }

    void pushBack(int val) {
    q.push_back(val);
    }

    int popFront() {
    if(q.empty())
        return -1;
    int popped = q.front();
    q.pop_front();
    return popped;
    }

    int popMiddle() {
    if(q.empty())
        return -1;
    int mid = (q.size()-1)/2;
    auto middle = q.begin() + mid;
    int popped = *middle;
    q.erase(middle);
    return popped;
    }

    int popBack() {
    if(q.empty())
        return -1;
    int popped = q.back();
    q.pop_back();
    return popped;
    }
};
```

Problem 5: Linked Lists (Difficulty: Medium)

Given the head of a linked list, rotate the list to the right by k places.

Solution Code:

```
#include<bits/stdc++.h>
using namespace std;

struct ListNode {
     int val;
     ListNode *next;
     ListNode() : val(0), next(nullptr) {}
     ListNode(int x) : val(x), next(nullptr) {}
     ListNode(int x, ListNode *next) : val(x), next(next)
{}
};

class Solution {
public:
     int length(ListNode* head)
     {
     ListNode* ptr = head;
     int len = 0;
     while(ptr)
     {
          len++;
          ptr = ptr->next;
     }
```

```
    return len;
    }
    ListNode* rotateRight(ListNode* head, int k) {
    if (head == nullptr)
        return nullptr;
    k = k%length(head);
    for(int i = 0; i < k; i++)
    {
        ListNode* ptr = head;
        while(ptr->next->next)
        {
            ptr = ptr->next;
        }
        ListNode* last = ptr->next;
        ptr->next = NULL;
        last->next = head;
        head = last;
    }
    return head;
    }
};
```

Links:

Github repository (Contains elaborate problem statements along with README description and code):
https://github.com/themohitnair/leetISL_36

Documentation used:
https://cplusplus.com/