ANSWER KEY

Stacks & Queues

Q1:

Similar problem:

https://www.ideserve.co.in/learn/sum-of-two-linked-lists-using-stacks

Q2:

Leetcode 20. Valid Parentheses

```
class Solution {
public:
    bool isValid(string s) {
        stack<char> bracketStack; //Keep track of open brackets
        //Iterate through the characters in the input string
        for (char c : s) {
            if (c == '(' || c == '[' || c == '{') {
               //If open bracket found push to stack
                bracketStack.push(c);
            } else {
                //If closing bracket found, check if stack
                //is empty or if it matches the top element
           }
If(bracketStack.empty()||!isMatching(bracketStack.top(), c)) {
                    //Mismatched or extra closing bracket
                    return false;
                }
                //Remove matching open bracket
                bracketStack.pop();
            }
        }
        //Check if there are any open brackets left in the stack
        return bracketStack.empty();
    }

private:
    bool isMatching(char open, char close) {
        return (open == '(' && close == ')') || (open == '[' &&
close == ']') || (open == '{' && close == '}');
    }
};
```

- https://leetcode.com/problems/valid-parentheses/solutions/4122752/beats-100-runtime-optimised-code-easy-solution-with-o-n-approach-using-stack-data-structure/
- https://leetcode.com/problems/valid-parentheses/

Q3:

Leetcode 225. Implement Stack using Queues

```
class MyStack {
public:
    queue<int> q1;
    MyStack() {

    }
    void push(int x) {
        int sz = q1.size();
        q1.push(x);
        while(sz--){
            q1.push(q1.front());
            q1.pop();
        }
    }
    int pop() {
        int result = top();
        q1.pop();
        return result;
    }
    int top() {
        return q1.front();
    }
    bool empty() {
        return q1.empty();
    }
};
```

- https://leetcode.com/problems/implement-stack-using-queues/solutions/3969698/3-easy-c-solutions-using-1-queue-and-2-queues-beats-100/
- https://leetcode.com/problems/implement-stack-using-queues/description/

Q4:

Leetcode 387. First Unique Character in a String

```
class Solution {
public:
    int firstUniqChar(string s) {
       queue<int> q;
       int arr[26]={0};
       for(int i=0;i<s.length();i++){
           char ch=s[i];
           q.push(i);
           arr[ch-'a']++;
           while(!q.empty()){
               if(arr[s[q.front()]-'a']>1)
                    q.pop();
                else break;
           }
       }
        if(q.empty())
            return -1;
        return q.front();
    }
};
```

- https://leetcode.com/problems/first-unique-character-in-a-string/solutions/3779340/2-methods-easy-clean-code/
- https://leetcode.com/problems/first-unique-character-in-a-string/

Q5: Write a program that can convert a postfix expression into an infix expression using a stack.

- A **postfix expression** is an expression where the operators are written after the operands such as: abc+++.
- An **infix expression** is where the operators are between operands. It's what we're already used to, so for example like: a + b + c.

ANSWER KEY

Old code from Rizk Fall 2021:
https://replit.com/@TariqAchorzyad/InfixPostfix-Program#expNotation.cpp
GeeksForGeeks: https://www.geeksforgeeks.org/postfix-to-infix/#

Linked Lists

Q1:
Pseudocode:
https://stackoverflow.com/questions/49872874/how-to-find-middle-element-of-doubly-linked-list-using-head-and-tail
Solution: https://kalkicode.com/find-middle-element-doubly-linked-list

Q2:
Leetcode 206. Reverse Linked List

- https://leetcode.com/problems/reverse-linked-list/solutions/3211778/using-2-methods-iterative-recursive-beats-97-91/
- https://leetcode.com/problems/reverse-linked-list/

Q3:
Leetcode 83. Remove Duplicates from Sorted List

- https://leetcode.com/problems/remove-duplicates-from-sorted-list/solutions/3257316/c-python-c-java-easiest-solution-o-n-time/
- https://leetcode.com/problems/remove-duplicates-from-sorted-list/

Q4:
Leetcode 203. Remove Linked List Elements

- https://leetcode.com/problems/remove-linked-list-elements/solutions/3086104/accepted-easy-solution-short-simple-best-method/
- https://leetcode.com/problems/remove-linked-list-elements/description/

Q5:

a. List and explain the types of Linked Lists you learned about in class.
   i. Singly linked, Doubly linked, Circularly linked
b. What does “traversal” of a Linked List mean? Explain the process.

   i. Traversing means visiting each node in the Linked List from the head to the tail. This is typically done by using a while loop and a pointer.

c. What are some advantages and disadvantages of Linked Lists compared to arrays?
   i. Linked List Advantages: Faster insertion and deletion, dynamic size, efficient memory allocation.
   ii. Linked List Disadvantages: Slow search times, requires more memory (lots of memory per nodes)

Heaps

Q1:

Min: {3, 8, 5, 15, 12, 9}

Max: {15, 12, 9, 3, 8, 5}

Q2:

[14, 13, 16, 8, 10]

Sorts

Show this array: A = {38, 27, 43, 3, 9, 82, 10}

After each pass of these sorts: Insertion, Selection, Mergesort, Quicksort

Insertion

Original: A = {38, 27, 43, 3, 9, 82, 10}

1. [27, 38, 43, 3, 9, 82, 10]
2. [27, 38, 43, 3, 9, 82, 10]
3. [3, 27, 38, 43, 9, 82, 10]
4. [3, 9, 27, 38, 43, 82, 10]
5. [3, 9, 10, 27, 38, 43, 82]

Sorted: A = {3, 9, 10, 27, 38, 43, 82}

Selection

ANSWER KEY

Original: A = {38, 27, 43, 3, 9, 82, 10}

1. [3, 27, 43, 38, 9, 82, 10]
2. [3, 9, 43, 38, 27, 82, 10]
3. [3, 9, 10, 38, 27, 82, 43]
4. [3, 9, 10, 27, 38, 82, 43]
5. [3, 9, 10, 27, 38, 43, 82]

Sorted: A = {3, 9, 10, 27, 38, 43, 82}

Mergesort

Original: A = {38, 27, 43, 3, 9, 82, 10}

1. [38, 27, 43] [3, 9, 82, 10]
2. [38] [27, 43] [3, 9] [82, 10]
3. [38] [27] [43] [3] [9] [82] [10]
4. [27, 38] [3, 43] [9, 82] [10]
5. [3, 27, 38, 43] [9, 10, 82]
6. [3, 9, 10, 27, 38, 43, 82]

Sorted: A = {3, 9, 10, 27, 38, 43, 82}

Quicksort

Original: A = {38, 27, 43, 3, 9, 82, 10}

Picked first value as pivot

1. [38] [27, 3, 9, 10] [43, 82]
2. [27] [3, 9, 10] [38] [43, 82]
3. [3] [9, 10] [27] [38] [43, 82]
4. [9] [10] [3] [27] [38] [43] [82]
5. [3, 9, 10] [27, 38, 43, 82]
6. [3, 9, 10, 27, 38, 43, 82]

Sorted: A = {3, 9, 10, 27, 38, 43, 82}