Problem 1: Stack - Sort Values in Stack

Given a stack of integers, write a function that sorts the stack using only push and pop operations. You can use an additional temporary stack but not any other data structure (like an array).

The top of the stack represents the smallest number in the stack.

Example:

> Input: `[34, 3, 31, 98, 92, 23]`
> Output: `[3, 23, 31, 34, 92, 98]`

Hint: Sort the stack by transferring elements between the original and a temporary stack, ensuring order is maintained.

Problem 1: Solution

```
void sortStack(Stack& stack) {
    Stack tempStack;
    while (!stack.isEmpty()) {
        int temp = stack.pop();
        while (!tempStack.isEmpty() && tempStack.peek() < temp) {
            stack.push(tempStack.pop());
        }
        tempStack.push(temp);
    }

    // Move the elements back to the original stack
    while (!tempStack.isEmpty()) {
        stack.push(tempStack.pop());
    }
}
```

> I have included the stack class implementation just in case you decide to share the solution with the class. However, the primary focus of the quiz is the `void sortStack(Stack& stack)` function, which is the solution that students need to write.

```
#include <iostream>

using namespace std;

class Stack {
private:
    class Node {
```

```
    public:
        int value;
        Node* next;

        Node(int value, Node* next) : value(value), next(next) {}
    };

    Node* top;

public:
    Stack() : top(nullptr) {}

    ~Stack() {
        while (top != nullptr) {
            Node* temp = top;
            top = top->next;
            delete temp;
        }
    }

    bool isEmpty() {
        return top == nullptr;
    }

    int peek() {
        if (isEmpty()) {
            cout << "Stack is empty" << endl;
            return -1;
        }
        return top->value;
    }

    void push(int value) {
        Node* newNode = new Node(value, top);
        top = newNode;
    }

    int pop() {
        if (isEmpty()) {
            cout << "Stack is empty" << endl;
            return -1;
        }
        Node* tempNode = top;
        int value = top->value;
        top = top->next;
        delete tempNode;
        return value;
    }

    void print() {
```

```
        Node* current = top;
        while (current != nullptr) {
            cout << current->value << " ";
            current = current->next;
        }
        cout << endl;
    }
};

int main() {
    Stack stack;
    stack.push(34);
    stack.push(3);
    stack.push(31);
    stack.push(98);
    stack.push(92);
    stack.push(23);

    sortStack(stack);

    cout << "Sorted stack: ";
    stack.print();

    return 0;
}
```