2.1 Question 1: Find Middle Element

Size: n

**Important parts of this question:*

Don’t change the stack/If it is empty: put std::underflow error

~~~
template <typename T>
T StackQuestions T ::findMiddle() const {
    if (this->isEmpty()) {
        throw std::underflow_error("Stack is empty");
    }
    size_T n = this size();
    size_T midIndex = n / 2;   // works with odd and even numbers
    //    return data[midIndex];
}
```cpp
template <typename T>
T StackQuestions<T>::findMiddle() const {
    if (stack.empty()) {
        throw std::underflow_error("Stack is empty");
    }

  std::stack<T> temp = stack;
    size_t n = temp.size();
    size_t midIndex = n / 2;

Output of T;
    for (size_t i = 0; i <= midIndex; i++) {
        result = temp.top();
        temp.pop();
    }
    return result;
}
~~~

2.2 Question 2: Reverse Stack

a-) reverse

b-) insertAtBottom

First one:

```
// Reverse the stack
void reverseStack(stack<int>& st) {
    if (st.empty()) return;

    int temp = st.top();
    st.pop();
    reverseStack(st);
    insertAtBottom(st, temp);
```

Second one:

```
// Insert an element to the bottom of stack
void insertAtBottom(stack<int>& , int x) {
    if (st.empty()) {
        st.push(x);
        return;
    }
    int temp = st.top();
    st.pop();

    insertAtBottom(st, x);
    st.push(temp);
}
```

***Using temporary stack to traverse:

```
Stack<t>temp(this);
```

3.1 Question 3: Split Alternate

using

```
template <typename T>

2 void splitAlternate(Stack<T>& source, Stack<T>& stack1, Stack<T>& stack2)


Stack<int> source;
source.loadFromJStack("majorstack");

// Source (top to bottom): 9,8,7,6,5,4,3



Stack<int> stack1, stack2;

StackUtils::splitAlternate(source, stack1, stack2);

// stack1 (top to bottom): 9,8,7,6

8 // stack2 (top to bottom): 5,4,3

9 // source stack is empty rn
```

3.2 Question 4: Simplify Path

Using

```
std::string simplifyPath(const std::string& path);
```

?????