Stack-based solution (write down, if any, assumptions about the problem):

Think about: What should each element be? What should be pushed onto the stack? When an element is popped, what should be done?

Assume elements are stored in an array (arr) of length arr.length.

shifts one position left

element Temp

// Loads stack

for (int i = arr.length - 1; i >= 0; i--)
    push (arr[i]);

3

Temp = top;
pop();

for (int i = 0; i < arr.length; i++)
    arr[i] = top;
pop();

3

arr[arr.length - 1] = temp.
Stack-based solution (write down, if any, assumptions about the problem):

Think about: What should each element be? What should be pushed onto the stack? When an element is popped, what should be done?

```c
int VentureSum(int value) {
    push value;
    while ((top() / 10) \equiv 0) {
        int temp = top(); pop();
        push (temp % 10); \checkmark
        push (temp / 10); \checkmark
    }
    while (true) {
        int temp1 = top(); pop();
        if (stack == empty) {
            return temp1; \checkmark
        } else {
            int temp2 = top(); pop();
            push (temp1 + temp2); \checkmark
        }
    }
}
```
Stack-based solution (write down, if any, assumptions about the problem):

Think about: What should each element be? What should be pushed onto the stack? When an element is popped, what should be done?

Assumption: we have a heap of a tree, already in the stack. 

```java
    int count = 1;
    while (!is.Empty())
        for (int i = 0; i < count; i++)
            if (!is.Empty())
                print top();
                pop();
            count = count * 2;
```
Stack-based solution (write down, if any, assumptions about the problem):

Think about: What should each element be? What should be pushed onto the stack? When an element is popped, what should be done?

```
while length() > 1
    Push 1st letter in stack 1 and remove first character
    Push last letter in stack 2 and remove last character

while stack 1 and stack 2 are not empty

    If top stack 1 = top stack 2
        pop stack 1
        pop stack 2
    else
        return false;

return true;
```
Team Name: blank

Stack-based solution (write down, if any, assumptions about the problem):

Think about: What should each element be? What should be pushed onto the stack? When an element is popped, what should be done?

```c
push(n % 2);
while (n != 0) {
    n = n / 2;
    push(n % 2);
}
while (!stack.empty()) {
    pop(top);
    Pop 1;
}
```
Stack-based solution (write down, if any, assumptions about the problem):

Think about: What should each element be? What should be pushed onto the stack? When an element is popped, what should be done?

Push elements of 2 lists into 2 stacks

temp top1 = stack1.top
while (Stack1 not empty)

    temp top2 = stack2.top
    while (Stack2 not empty)
        temp2 = stack2.top

    if (temp1 == temp2)
        print temp1

        temp top2 = -;
    
        temp top1 = -;

This is ambiguous handling of temp top2 and temp top1.