1) The stack would have to be moved into the queue two time. The first time, check each pop of the stack to see if it is the particular value we are looking for. After the first time the stack should be backwards on the stack. If we repeat the process for a second time the stack would be back to it’s original order.

2) The recursive function takes in the string and the length of the string. Each time it runs it returns a number plus another recursive call, but each time the length is decreased by 1. The number the function returns would be computer with (temp \* pow(10,string.length() - len)), where temp is the integer form of string[len-1] and len is the length of the string. The base cause would just be when len reaches 0 and returns 0;

3) #include <iostream>

#include <iterator>

using namespace std;

int main() {

Node<T>\* next;

Node<T>\* curr = \_head;

Node<T>\* prev = nullptr;

for(Node<T>\* itr = \_head;itr != nullptr;itr = itr->\_next){

next = curr\_next;

curr\_next = prev;

prev = curr;

curr = next;

}

}

4) The forloop goes through the linked list one time doing an operation on each node one time.

5) ADTs are mainly defined by the operations they do and not how they are implemented, while data structures are defined by the way they are organized and the form of how it is organized. ADTs are implemented using data structures. Examples of ADTs are lists, stacks, and queues and all these could be implemented using a linked list.