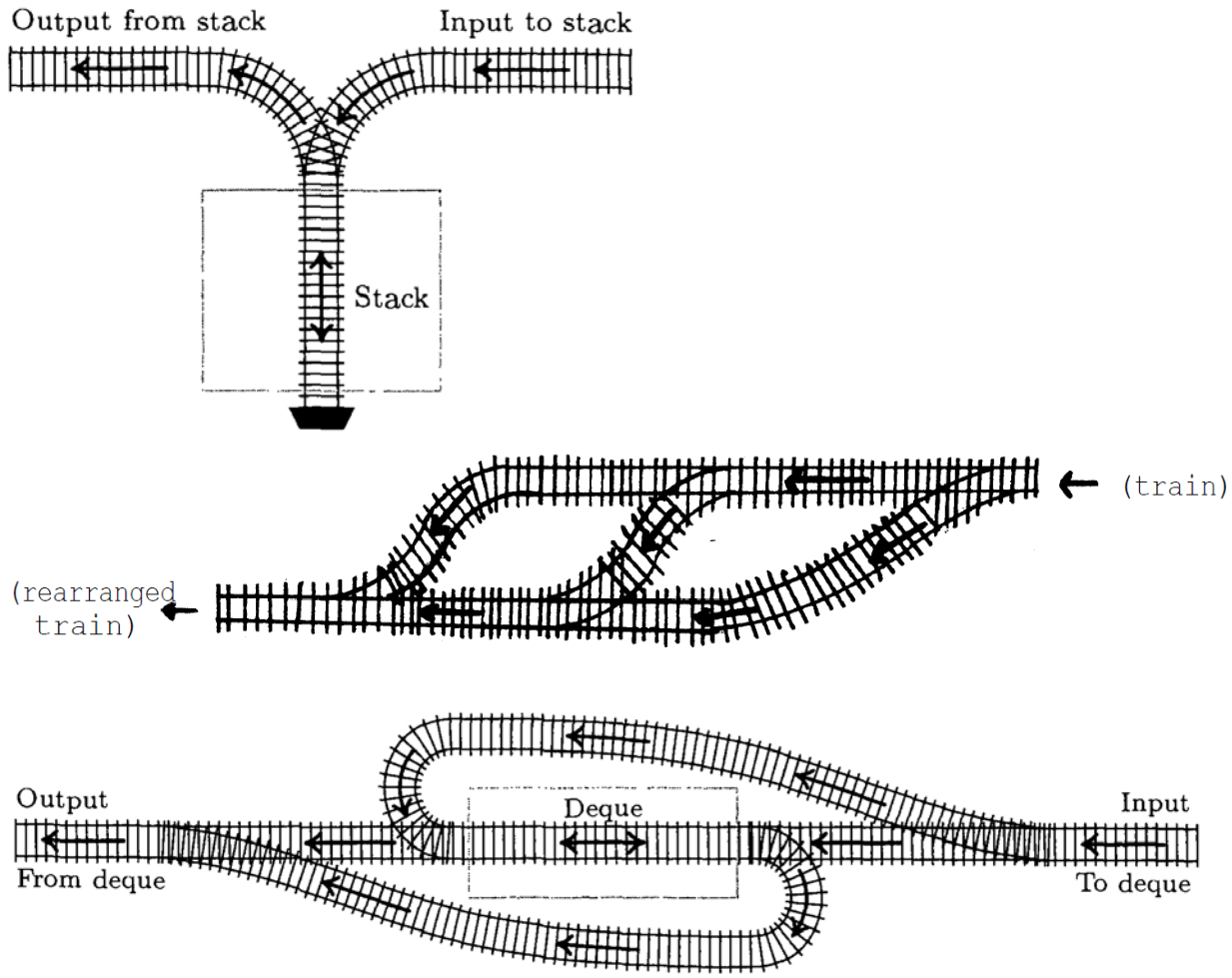
CS 101 **Lab 3** **Summer 2017**

Trains in a railroad switchyard can be used to model data structures such as stacks, queues, and deques. Cars can be decoupled from one train, and then recoupled to either the same train or a different train.



Implement a C++ class Train that simulates the sequence of cars in a train. Each car is labeled with an id number. Your class Train must provide these operations:

- Train(s, n) constructs a new train that contains n sequentially numbered cars from s at the front up to s+n–1 at the back.
- display( ) prints the numbers of the cars in this train from front to back.
- reverse( ) re-orients this train in the opposite direction.
- move(k, t) detaches k cars from the front of this train and attaches them to the back of train t.

Your class should behave as shown in the following example. Make sure you understand the definition of each operation before you start to write its code.

int main( ) {
  Train t1(10, 8);
  Train t2(20, 5);
  cout << "t1: "; t1.display( );
  cout << "t2: "; t2.display( );
  t1.reverse( );
  t1.move(4, t2);
  cout << "t1: "; t1.display( );
  cout << "t2: "; t2.display( );
  t2.move(2, t1);
  t2.reverse( );
  cout << "t1: "; t1.display( );
  cout << "t2: "; t2.display( );
  return 0;
}
	Output
	
	t1: 10 11 12 13 14 15 16 17
	t2: 20 21 22 23 24
	t1: 13 12 11 10
	t2: 20 21 22 23 24 17 16 15 14
	t1: 13 12 11 10 20 21
	t2: 14 15 16 17 24 23 22

Please carefully read the following requirements:

- Try to write the program individually, because you will learn more that way. However, if you get stuck and don't know how to proceed, you can ask the instructor or a classmate for assistance.
- You may demonstrate your program either on your local machine or on the cs-intro.ua.edu server.
- Once you believe your program runs correctly using all the above examples, demonstrate your program to the instructor. If the instructor agrees that your program works correctly, you are done.