**Solution**

**Problem1:**

Firstly, I write two “.h” files for the Node and the LinkedStack. And implement methods of LinkedStack class in the file called “LinkedStack.cpp”.

For the constructor of the LinkedList, I initiate a new node called first and let the first points to null.

For the destructor, I initiate the pointer called “current” whose type is node and let the pointer points to the first node. Then I initiate a new pointer called “to\_free”. While using “current” pointer to traverse the linked list, the “to\_free” pointer will point to the node before “current” pointer. And then, use the delete method to destroy the node. As a result, it will destroy the whole nodes.

For the push() method, I initiate the node pointer called “conductor”. And let the conductor pointer points to the last node of the stack. Then, I initiate the new node storing the data with its next points to null. And let the conductor’s next points to the new node.

For the top() method, to begin with, I initiate a new node pointer called “conductor” and let it points to the first. Then, using empty() method, if the empty() returns true, then output the stack is empty. Otherwise, using the “conductor” point the traverse all node and returns the data part of the last node.

For the empty() method, I just estimate that whether the next part of the first node points to another node. If it points to another node, it indicates that the stack is not empty. Otherwise, it is empty.

For the pop() method, I initiate two node pointers called ”conductor” and “previous”. The previous pointer always points to the node before the conductor points. Then traverse all nodes to let the “conductor” pointer points to the last node. Then return the data part of the node which “conductor” pointer points and set the next part of the node which “previous” pointer points to null.

**Problem2:**

In the Railroad() method, firstly, I use a vector whose type is LinkedStack to store the holding stacks. I use the numberofTracks to set the size of the vector. Then do loops to deal with the input cars according to the number of cars. In the meantime, I also use a target number to indicate the number that should be output next. While traversing the input array, if the current number is not equal to the target number, then I will use the putinHold() method to put the number into one stack. In the putinHold() method, I traverse all the stack and get numbers on the top of the stack from left to right. If the number, which is going to input, is small then the top number I get from one stack, then I will push the number into current stack. Otherwise, I will go to the next stack. If the number id bigger than any top number, then I will store that number into a new holding stack and back to the Railroad() method to deal with the next number in the input array. If the current number is equal to the target number, then I will output it directly. After that, I will use the outputFromTrack() method to see can any number from stack be output. In the outputFromTrack() method, I simply get all top number from all stacks to estimate whether any one is equal to the target number. If there is no equal, then I will deal with the next number from input. Otherwise, I will output the eligible number and search all tracks again to find out the next target number.

**Problem3:**

To read data from input.txt file, I use ifstream method to get the data from txt file. For the input array, due to we do not know the size of it while initiate the array, I initiate a int type pointer.

For print message while dealing with the input array, I put the message of holding cars in putinHold() method. Dueing the process, I can record the number going to put into holding stack and the stack number. Therefore, I can output the holding message. I put the message of moving cars from input in Railroad() method. In the Railroad() method, if I find that the number in the array can be output directly, then I will show the message. I put the message of moving cars from holding tracks to output in the outputFromHoldTrack() method. If I find out that the number store in the top of holding tracks can be output. I can show the message according to the stack number.

**Problem4:**

For solving this problem, I initiate a global variable called “success”. At the end of the procedure, if this program can output the sorted input successfully, then I will assign true to that variable. Otherwise, I will assign false. Besides, I store all messages into a vector. While storing the message, due to the value of the number and the number of stacks are int types and they cannot be added into the string message directly. Then, I change all these int numbers into string type by using stringstream method. Finally they can be added into the string message successfully. If the procedure is successful, then I will traverse that vector to show all messages. Otherwise, I will show the numbers of input and plus the words “is not feasible”.

**Problem5:**

The permutation of cars are 1, n, n-1, n-2, … ,2.

The reason is that if we want to use n-1 holding stack. We should put the first number, namely 1, at the back of the input array. For the rest permutation of the number we should let the previous input number become smaller than the current input number. Therefore, the current input number cannot be stored at the top of existed stacks, so we have to input the current input number into a new holding stack.

**Problem6:**

To