Firstly, I write two “.h” files for the Node and the LinkedStack. I write the variables and methods of these two classes in corresponding files. Then I implement method 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 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.