Question 1:

Because this is C code, so we do not have template to use, but we can use #define to change the type value_type;

We need two dummy nodes (head and tail): head->node_1->... node_n->tail,

1. class Node: Node* next: a pointer point to its next node in the list;
2. class DataNode:
   (1) value_type data: Value of the data stored in the node, which type is value_type;
   (2) DataNode* next: a pointer point to its next node in the list;
3. iterator : a pointer to a node, and have following functions:
   (1) & : dereference of iterator;
   (2) ++it or it++ : increment of iterator;
   (3) it1<it2,it1>it2 : comparison of iterators;
   (4) it1=it2 assignment of iterators;
4. (1) begin() : Return iterator to head node
   (2) end() : Return iterator to tail node
5. iterator insert(iterator position, const value_type & value)
   (1) Function meaning:
   Create a new node that contains the data given in the second parameter and insert it in the list in the pos before the node that the first parameter points to.
   We can also use insert(end(), value), to insert a data node in the end of the list;
   (2) Returns value: An iterator that points to the new node;
6. iterator erase(iterator position)
   (1) Function meaning:
   Removes a node from the location pointed to by the iterator in its parameter;
   Because we erase by position, so we do not need to worry about dup elements;
   (2) Returns value: Returns an iterator that points to the node after the removed node.

Question 2:

1. Test for class Node : check if it works normally, like access to its *next;
2. Test for class DataNode: : check if it works normally, like access to the data and *next;
3. Test for iterators: like &, ++, comparison, assignment;
4. Test for begin(): should return iterator that point to the head of list;
4. Test for end(): should return iterator that point to the tail of list;
5. Test for insert: check if it insert the new DataNode correctly, and return value;
   Then check if the list can be traversed correctly;
6. Test for erase: check if it erase DataNode in the selected pos correctly, and return value;
   Then check if the list can be traversed correctly;

Question 3:

Using pthread_mutex to make sure only one thread are using the list during insert and erase operations at the same time;

Before insert or erase operations, set a lock: pthread_mutex_lock(mutex);:

After insert or erase, unlock and destroy the lock;

pthread_mutex_unlock(mutex); pthread_mutex_destroy(mutex);