```
1  #include <stdlib.h>
2  #include <pthread.h>
3
4  struct node {
5      pthread_mutex_t mutex;
6      struct node *next;
7      int value;
8  };
9
10 struct node head = {
11     .mutex = PTHREAD_MUTEX_INITIALIZER,
12     .next = NULL,
13     .value = 0,
14 };
15
16 void insert(int value) {
17     struct node *node = &head;
18     struct node *next;
19
20     pthread_mutex_lock(&node->mutex);
21     while ((next = node->next)) {
22         pthread_mutex_unlock(&node->mutex);
23         pthread_mutex_lock(&next->mutex);
24         if (next->value < value) {
25             node = next;
26         } else {
27             pthread_mutex_unlock(&next->mutex);
28             pthread_mutex_lock(&node->mutex);
29             break;
30         }
31     }
32
33     next = malloc(sizeof(*next));
34     pthread_mutex_init(&next->mutex, NULL);
35     next->value = value;
36     next->next = node->next;
37     node->next = next;
38     pthread_mutex_unlock(&node->mutex);
39 }
```

The code on the left is supposed to implement thread-safe insertion into a sorted linked list. Suppose two threads run concurrently:

Thread 1: `insert(1); insert(3); insert(5);`
Thread 2: `insert(6); insert(4); insert(2);`

Question 1. Does the code contain any data races? If so, how would you fix them?

Question 2. Does the code work as intended? If not, how would you fix it?

Question 3. If a thread is preempted at line 22 in the middle of the list, can the other thread insert at the beginning of the list? What about the end of the list?