Pthreads 作业：任务队列

介绍

编写一个 Pthreads 程序实现“任务队列”(Task queue)，主线程负责启动用户指定数量的线程，这些线程会因为需要等待条件触发而立即睡眠。同时，主线程还生成由其他线程执行的任务（任务内容由任务选项指定，而任务选项是[0,5]范围内随机的数字）；每次它生成一个新任务块，就会用一个条件信号唤醒一个线程。当一个线程完成任务块的执行时，又回到条件等待状态。主线程完成所有的生成任务后，会设置某个全局变量，以指示再也没有新的任务可以生成了，并用一个条件广播唤醒所有的线程。任务采用链表操作。

要求

编译：

```
$ gcc -g -Wall -o taskqueue taskqueue.c -lpthread
```

命令行运行，后面分别为线程数和生成任务数：

```
$ ./taskqueue <number of threads> <number of tasks>
```

输出任务执行的过程：

每个任务打印：“Thread 线程 ID: task 任务 ID:” + 任务内容，参照样例。

最后一行打印所有链表元素。

注意：

- 任务是对元素为 int 类型链表的相关操作，每个任务根据任务选项进行不同的操作，如下
- 任务选项是一个[0,5]中的随机数字：

  0：将一个随机整数插入链表
  - 若链表已存在这个数字则插入失败，输出“%d cannot be inserted”
  - 插入成功并输出“%d is inserted”

  1：删除链表中随机一个整数
  - 若链表中不存在这个数字则删除失败，输出“%d cannot be deleted”
  - 删除成功并输出“%d is deleted”

  2：查看数据是否在链表内
  - 若在链表内输出“%d is in the list”
  - 若不在链表内则输出“%d is not in the list”

  default：若不是上述选项，则输出“print list:”并在其后打印目前链表的所有元素
- 链表使用一个互斥量(mutex)保护而不用读写锁
- 任务数尽可能为线程数的倍数

整数链表和任务队列参照

```
/* Struct for list nodes */
struct list_node {
   int    data;
   struct list_node* next;
};

/* Struct for task nodes */
struct task_node {
   int task_id;
   int task_option;
   int data;
   struct task_node* next;
};
```

样例 A

(任务的顺序和内容是随机的)

```
$ ./taskqueue 4 16
Thread 0: task 0: 3 is inserted
Thread 3: task 1: 17 is inserted
Thread 1: task 2: 13 is inserted
Thread 1: task 6: print list: 3 13 17
Thread 1: task 7: 3 cannot be inserted
Thread 1: task 8: 0 is inserted
Thread 1: task 9: 12 is inserted
Thread 1: task 10: 11 is not in the list
Thread 1: task 11: print list: 0 3 12 13 17
Thread 1: task 12: 2 is inserted
Thread 1: task 13: 2 is in the list
Thread 1: task 14: 7 is inserted
Thread 1: task 15: 9 cannot be deleted
Thread 0: task 4: 9 is inserted
Thread 3: task 5: 2 is deleted
Thread 2: task 3: 6 cannot be deleted
Final list:
   0 3 7 9 12 13 17
```

样例 B

```
$ ./taskqueue 4 4
Thread 0: task 0: 3 is inserted
Thread 2: task 1: 17 is inserted
Thread 1: task 2: 13 is inserted
Thread 3: task 3: 6 cannot be deleted
Final list:
   3 13 17
```

样例 C

```
$ ./taskqueue 4 8
Thread 1: task 0: 3 is inserted
Thread 1: task 4: 9 is inserted
Thread 1: task 5: 2 cannot be deleted
Thread 1: task 6: print list: 3 9
Thread 0: task 1: 17 is inserted
Thread 2: task 2: 13 is inserted
Thread 3: task 3: 6 cannot be deleted
Thread 1: task 7: 3 cannot be inserted
Final list:
   3 9 13 17
```