线程测验

1. 写出实现简单线程池的伪代码，包括线程池的数据结构，初始化操作，添加任务操作，销毁操作。

2. 哲学家就餐问题

五个哲学家共用一张圆桌，分别坐在周围的五张椅子上，在圆桌上有五个碗和五只筷子，他们的生活方式是交替的进行思考和进餐。

平时，一个哲学家进行思考，饥饿时便试图取用其左右最靠近他的筷子，只有在他拿到两只筷子时才能进餐。进餐完毕，放下筷子继续思考。

设计一个程序五个哲学家共用一张圆桌，分别坐在周围的五张椅子上，在圆桌上有五个碗和五只筷子，他们的生活方式是交替的进行思考和进餐。平时，一个哲学家进行思考，饥饿时便试图取用其左右最靠近他的筷子，只有在他拿到两只筷子时才能进餐。进餐完毕，放下筷子继续思考。

下面为此问题的一种代码实现，请根据自己的理解写出代码中的思路，以注释的形式加在每步：

```
#include <stdio.h>
#include <stdlib.h>
#include <memory.h>
#include <pthread.h>
#include <errno.h>
#include <unistd.h>
#include <math.h>
pthread_mutex_t chopstick[6] ;
void get(int *left,int *right,char phi) {
    switch (phi){
        case 'A':
            *left = 5;
            *right = 1;
            break;
        case 'B':
            *left = 1;
            *right = 2;
            break;
        case 'C':
            *left = 2;
            *right = 3;
            break;
        case 'D':
            *left = 3;
            *right = 4;
            break;
        case 'E':
            *left = 4;
            *right = 5;
            break;
    }

```

```
33
34  }
35  void *eat_think(void *arg)
36  {
37      char phi = *(char *)arg;
38      int left,right;
39      get(&left,&right,phi);
40      for(;;){
41          usleep(10);
42          pthread_mutex_lock(&chopstick[left]);
43          printf("Philosopher %c fetches chopstick %d\n", phi,
    left);
44          if (pthread_mutex_trylock(&chopstick[right]) ==
    EBUSY){
45              pthread_mutex_unlock(&chopstick[left]);
46              continue;
47          }
48
49          printf("Philosopher %c fetches chopstick %d\n", phi,
    right);
50          printf("Philosopher %c is eating.\n",phi);
51          usleep(10);
52          pthread_mutex_unlock(&chopstick[left]);
53          printf("Philosopher %c release chopstick %d\n", phi,
    left);
54          pthread_mutex_unlock(&chopstick[right]);
55          printf("Philosopher %c release chopstick %d\n", phi,
    right);
56
57      }
58  }
59  int main(){
60      pthread_t A,B,C,D,E; //5个哲学家
61
62      int i;
63      for (i = 0; i < 5; i++)
64          pthread_mutex_init(&chopstick[i],NULL);
65      pthread_create(&A,NULL, eat_think, (void *)"A");
66      pthread_create(&B,NULL, eat_think, (void *)"B");
67      pthread_create(&C,NULL, eat_think, (void *)"C");
68      pthread_create(&D,NULL, eat_think, (void *)"D");
69      pthread_create(&E,NULL, eat_think, (void *)"E");
70
71      pthread_join(A,NULL);
72      pthread_join(B,NULL);
73      pthread_join(C,NULL);
74      pthread_join(D,NULL);
75      pthread_join(E,NULL);
76      return 0;
77  }
```

3. 编程实现生产者消费者模型，生产者的主要作用是生成一定数量的数据放到缓冲区中，然后重复此过程。

   与此同时，消费者也在缓冲区消耗着这些数据。

   要求：保证生产者不会在缓冲区满时加入数据，消费者也不会在缓冲区中空时消耗数据。

   提示：缓冲区可以设置为全局数组,生产者和消费者均为一个线程,利用线程间同步机制实现。