```c
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
#include "cond.c"


int pnum;  // number updated when producer runs.
int csum;  // sum computed using pnum when consumer runs.

pthread_mutex_t mymutex;
pthread_cond_t mycondvar;

int (*pred)(int); // predicate indicating if pnum is to be consumed

int produceT() {
  scanf("%d",&pnum); // read a number from stdin
  return pnum;
}

void *Produce(void *a) {
        int p;
          p = 1;

        pthread_mutex_lock(&mymutex);
          while (p) {
                pthread_cond_signal(&mycondvar);
//              printf("unlocks and sleeps\n");

                printf("@P-READY\n");
                    p = produceT();
                    printf("@PRODUCED %d\n",p);

                pthread_cond_wait(&mycondvar, &mymutex);
//              printf("wakes up and locks\n");
          }
        pthread_mutex_unlock(&mymutex);
//        printf("Produce wakes up & unlocks mymutex\n");

          printf("@P-EXIT\n");
          pthread_exit(NULL);
}


int consumeT() {
        if (pred(pnum)) {
        csum += pnum;
        }
          return pnum;
}

void *Consume(void *a) {
          int p;
        p = 1;

        pthread_mutex_lock(&mymutex);
//        printf("Consume has locked mymutex\n");
        while (p) {

                pthread_cond_signal(&mycondvar);
//                printf("tells Produce to wake up\n");

                printf("@C-READY\n");
                    p = consumeT();
                    printf("@CONSUMED %d\n", csum);

                if (p != 0)
                        pthread_cond_wait(&mycondvar, &mymutex);

        }
        pthread_mutex_unlock(&mymutex);
//        printf("Consume has unlocked my mutex\n");

          printf("@C-EXIT\n");
          pthread_exit(NULL);
}


int main (int argc, const char * argv[]) {
          // the current number predicate
          static pthread_t prod,cons;
        long rc;
```

```
          pred = &cond1;
          if (argc>1) {
                    if (!strncmp(argv[1], "2", 10)) {
                        pred = &cond2;
                }
                    else if (!strncmp(argv[1], "3", 10)) {
                        pred = &cond3;
                }
        }

        pthread_cond_init(&mycondvar, NULL);
        pthread_mutex_init(&mymutex, NULL);

          pnum = 999;
          csum = 0;
          srand(time(0));

          printf("@P-CREATE\n");
         rc = pthread_create(&prod, NULL, Produce, (void *)0);
        if (rc) {
                printf("@P-ERROR %ld\n",rc);
                exit(-1);
        }

        printf("@C-CREATE\n");
         rc = pthread_create(&cons, NULL, Consume, (void *)0);
        if (rc) {
                printf("@C-ERROR %ld\n", rc);
                exit(-1);
        }

          printf("@P-JOIN\n");
          pthread_join(prod, NULL);
          printf("@C-JOIN\n");
          pthread_join(cons, NULL);


          printf("@CSUM=%d.\n",csum);

         return 0;
}
```