```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
#include <errno.h>

#define NUM_OF_NODES 10

static pthread_mutex_t mutex;
static pthread_cond_t cond;

struct node {
      int n_number;
      struct node *n_next;
} *head;

static void *thread_func(void *arg) {
        int err;
        struct node *p;
        int i;

        for(i=0; i<NUM_OF_NODES; i++) {
                if (err = pthread_mutex_lock(&mutex)) { /* lock mutex */
                        printf("pthread_mutex_lock: %s\n",strerror(err));
                        exit(1);
                }
                // check if queue is empty
                if(head == NULL)
                  // wait producer to put a node in the queue
                        if (err = pthread_cond_wait(&cond, &mutex)) {
                                printf("pthread_cond_wait: %s\n",strerror(err));
                                exit(1);
                        }

            // modify shared variable head
                  p=head;
                  head=head->n_next;
                  printf("Elava %d apo tin arxi tis ouras\n", p->n_number);
                  // free memory associated with node that has been just
dequeued
                  free(p);
                  if (err = pthread_mutex_unlock(&mutex)) { /* unlock mutex */
                        printf("pthread_mutex_unlock: %s\n",strerror(err));
                        exit(1);
                  }
        }
        return (void *)1;
}

int main() {
        pthread_t tid;
        void *status;
        int err, i;
        struct node *p;
        pthread_mutex_init(&mutex, NULL);
      pthread_cond_init(&cond, NULL);

        /* create thread */
        if(err=pthread_create(&tid, NULL, thread_func, NULL)) {
                printf("pthread_create: %s\n",strerror(err));
                exit(1);
```

```
        }

        /* put nodes in the queue  */
        for(i=0; i<NUM_OF_NODES; i++) {
                p = malloc(sizeof(struct node));
                p->n_number = i;

                if (err = pthread_mutex_lock(&mutex)) { /* lock mutex */
                        printf("pthread_mutex_lock: %s\n",strerror(err));
                        exit(1);
                }
                //shared variable
                p->n_next = head;
                head = p;

                /* send signal to condition  */
                if (err = pthread_cond_signal(&cond)) {
                        printf("pthread_cond_signal: %s\n",strerror(err));
                        exit(1);
                }
                if (err = pthread_mutex_unlock(&mutex)) { /* Unlock Mutex */
                        printf("pthread_mutex_unlock: %s\n",strerror(err));
                        exit(1);
                }

                sleep(1);
        }

        if(err=pthread_join(tid, &status)) {
                printf("pthread_join: %s\n",strerror(err));
                exit(1);
        }
      pthread_mutex_destroy(&mutex);
      pthread_cond_destroy(&cond);

        printf("H ergasia oloklirwthike. H katastasi exodou tou nimatos 2
einai %d\n", (int)status);
        return 0;
}
```