```
//input1
Design patterns are considered as an important element in computer
science. [DELAY] A design pattern describes a recurring design
problem, a solution, and the context in which that solution works.
[NEWLN] It also names a technique and describes its costs and
benefits. [DELAY]

Successful application frameworks are full of design patterns.
[ERASE}  Examples of design patterns can be found in frameworks.
[OVER]
//end of input1

//beginning of output for ./proj2 < input1
Design patterns are considered
 as an important element in co
mputer science.  A design patt
ern describes a recurring desi
gn problem, a solution, and th
e context in which that soluti
on works.
 It also names a technique and
 describes its costs and benef
its.
Error: non-displayable character

Error: non-displayable character
Successful application fr
ameworks are full of design pa
tterns.   Examples of design p
atterns can be found in framew
orks.
the end
//end of output for ./proj2 < input1
```

```
//input2
Design patterns are considered as an important element in computer
science. [DELAY] A design pattern [DBNLN] describes a [DELAY]
recurring design problem, a solution, and the context in which that
solution works. [NEWLN] It also names a technique[INDNT] and
describes its costs and benefits. [DELAY][EKNDC]

Successful application frameworks are full of design patterns.
[ERASE}  Examples of design patterns can be found in frameworks.
[OVER]
//end of input2

//beginning of output for ./proj2 < input2
Design patterns are considered
 as an important element in co
mputer science.  A design patt
ern

 describes a  recurring design
 problem, a solution, and the
context in which that solution
 works.
 It also names a technique
      and describes its costs
and benefits.
Error: unrecognized command

Error: non-displayable character

Error: non-displayable character
Successful appli
cation frameworks are full of
design patterns.   Examples of
 design patterns can be found
in frameworks.
the end
//end of output for ./proj2 < input2
```

```
CFLAGS = -Wall -pthread
EFILE = proj2
DEPS = proj2.h
OBJS = proj2.o thread1.o thread2.o

$(EFILE): $(OBJS)
	gcc $(CFLAGS) -o $(EFILE) $(OBJS)

%.o: %.c $(DEPS)
	gcc $(CFLAGS) -c $<
```

```
//proj2.h
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>

void * controller();
char bufferReader ();
void execComm ();
void displayText (char ch);
void * devDriver(void * arg);
void bufferWriter(char data[]) ;
```

```
//proj2.c
#include "proj2.h"

// mutex and condition variables to ensure correct access to shared
memory
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t reading_done = PTHREAD_COND_INITIALIZER;
pthread_cond_t writing_done = PTHREAD_COND_INITIALIZER;

// memory shared between threads
int current = 7;
char buffer[7];

int main()
{
     int rc1, rc2;
     int x = 1, y = 2;
     pthread_t thread1, thread2;

     /* create two threads, each runing a different function */

     if( (rc1=pthread_create( &thread1, NULL, controller, &x)) )
     {
               printf("Thread creation failed: %d\n", rc1);
     }

     if( (rc2=pthread_create( &thread2, NULL, devDriver, &y)) )
     {
               printf("Thread creation failed: %d\n", rc2);
     }

     /* wait till each thread is complete before main continues */

     pthread_join( thread1, NULL);
     pthread_join( thread2, NULL);

     printf("\nthe end\n");

     return 0;
}
```

```
//thread1.c
#include "proj2.h"

#include <time.h>
#include<stdio.h>
#include<stdlib.h>

extern pthread_mutex_t mutex;
extern pthread_cond_t reading_done;
extern pthread_cond_t writing_done;

extern int  current;
extern char buffer[];

void process(char);

int linepos = 0;

void * controller(void * arg)
{
    char ch;
    int j;

    char command[7];

    while(1) {
        if (linepos == 30){
            printf("\n");
            linepos = 0;
        }
        ch = bufferReader();
        if (ch!=EOF)
            displayText(ch);
        else
            break;
    }
     return 0;
}

void displayText (char ch) {
    if (isprint(ch))
        if (ch == '[')
            execComm();
        else{
            printf("%c", ch);
            linepos++;
        }
    else
        printf("\nError: non-displayable character\n");

}

void execComm (){
    char comm[5];
    int i;
    char ch;
    for (i = 0; i < 5; i++) {
        ch = bufferReader();
        comm[i] = ch;
    }
    if (bufferReader() == ']'){
        if (!strcmp(comm, "NEWLN")){
            printf("\n");
            linepos=0;
        }
        else if (!strcmp(comm, "DELAY")){
            sleep(1);
        }
        else if (!strcmp(comm, "DBNLN")){
            printf("\n\n");
            linepos = 0;
        }
        else if (!strcmp(comm, "INDNT")){
            printf("\n     ");
            linepos = 5;
        }
        else{
            printf("\nError: unrecognized command\n");
        }
    }
}


// bufferReader() returns one character from the shared buffer

char bufferReader()
{
     char ch;

     pthread_mutex_lock( &mutex );

     // wait if the next set of characters is not ready yet
     if (current == 7)
          pthread_cond_wait( &writing_done, &mutex );
     ch = buffer[current++];
     // signal to writer if the current set of characters is finished

     if (current == 7)
          pthread_cond_signal(&reading_done);

     pthread_mutex_unlock( &mutex );

     return ch;
}
```

```
//thread2.c
#include "proj2.h"

extern pthread_mutex_t mutex;
extern pthread_cond_t reading_done;
extern pthread_cond_t writing_done;

extern int  current;
extern char buffer[];

void * devDriver(void * arg)
{

    int i;
    int n = 0;
    char ch;
    char buff[7];

    while(1){
        for (i=0;i<7;i++){
            ch = getchar();
            if (ch!= EOF)
                buff[i] = ch;
            else {
                buff[i] = ch;
                bufferWriter(buff);
                return 0;
            }
        }
            bufferWriter(buff);
    }

     return 0;
}


// bufferWriter() stores the seven characters in array data into the

// shared buffer

void bufferWriter(char data[])
{
     pthread_mutex_lock(&mutex);

     // wait if the current set of characters is finished yet
     if (current < 7)
          pthread_cond_wait(&reading_done, &mutex);
    strncpy(buffer, data, 7);
    current = 0;
    // signal to reader when the next set of characters is ready
    pthread_cond_signal(&writing_done);

    pthread_mutex_unlock(&mutex);
}
```

```
commit 2ae36fb852956094fd2b3f52a345e92937446f13
Author: russjohnson09 <russjohnson09@gmail.com>
Date:   Thu Mar 21 19:38:36 2013 -0400

    final draft

commit 802c5e28ee5fd44bd261591a26895c46cc27b6ab
Author: russjohnson09 <russjohnson09@gmail.com>
Date:   Thu Mar 21 19:28:24 2013 -0400

    edit2

commit 28faf1a83d679263322db66cf22ae27b6303ad89
Author: russjohnson09 <russjohnson09@gmail.com>
Date:   Thu Mar 21 19:24:34 2013 -0400

    edit1

commit 8e91f69a6902a4bd819dc03f3f318afef08344ab
Author: russjohnson09 <russjohnson09@gmail.com>
Date:   Thu Mar 21 18:06:29 2013 -0400

    added gitignore

commit ec28e2f213f80530da698f0f7df9ce4f9de8f913
Author: russjohnson09 <russjohnson09@gmail.com>
Date:   Thu Mar 21 18:05:39 2013 -0400

    added make file

commit 7ed1f0429790b005ca3f23181ecfb90371937422
Author: russjohnson09 <russjohnson09@gmail.com>
Date:   Thu Mar 21 18:05:04 2013 -0400

    added input

commit 152836544652bb2a5f329c9de3915ed8a050910d
Author: russjohnson09 <russjohnson09@gmail.com>
Date:   Thu Mar 21 18:03:47 2013 -0400

    init
```