```
/* Readers/Writers Problem using Semaphores */
/* Writers have priority */

#include <stdio.h>
#include <conio.h>

typedef struct Semaphore { int count; } semaphore;

int readcount, writecount; semaphore x, y, z, wsem, rsem;

int wait(semaphore *s) {

	(*s).count--;

	if((*s).count<0) {
		return 0;
	}

	return 1;
}

void signal(semaphore *s) {
	(*s).count++;
}

void READUNIT() { printf("\tReader is reading.\n"); }
void WRITEUNIT() { printf("\tWriter is writing.\n"); }

void reader() {
	printf("\n\nReader:\n");
	if(!wait(&z)) { printf("\tReader cannot read. 'rsem' Queue is full\n") ; return; }
	if(!wait(&rsem)) { printf("\tReader cannot read. Writing in progress.\n"); return; }
	if(!wait(&x)) { printf("\tReader cannot read because readcount is locked\n");  return; }
	readcount++;
	if(readcount==1) wait(&wsem);
	signal(&x);
	signal(&rsem);
	signal(&z);
	READUNIT();
	if(!wait(&x))  { printf("\tReader cannot quit because readcount is locked\n");  return; }
	readcount--;
	if(readcount==0) signal(&wsem);
	signal(&x);
}

void writer(int r) {
	printf("\n\nWriter:\n");
	if(!wait(&y)) { printf("\tWriter cannot write because writecount is locked\n");  return;
}
	writecount++;
	if(writecount==1)
		wait(&rsem);
	signal(&y);
	if(!wait(&wsem)) {
		printf("\tWriter cannot write because book is locked.\n");
		return;
	}
	WRITEUNIT();
	signal(&wsem);
	if(!wait(&y)) { printf("\tWriter cannot quit because writecount is locked\n");  return; }
	writecount--;
	if(r==1) { reader(); signal(&z); signal(&rsem);}
	if(writecount==0)
		signal(&rsem);
	signal(&y);

}

void main() {

	x.count = y.count = z.count = wsem.count = rsem.count = 1;

	clrscr();

	writer(0);

	writer(1);

	reader();

	getch();

}
```

OUTPUT

```
Writer:
        Writer is writing.

Writer:
        Writer is writing.

Reader:
        Reader cannot read. Writing in progress.

Reader:
        Reader is reading.
```

```
/* Readers/Writers Problem using Semaphores */
/* Readers have priority */

#include <stdio.h>
#include <conio.h>

typedef struct Semaphore {
	int count;
} semaphore;

int readcount;
semaphore x, wsem;

int wait(semaphore *s) {

	(*s).count--;

	if((*s).count<0) {
		return 0;
	}

	return 1;
}

void signal(semaphore *s) {
	(*s).count++;
}

void READUNIT() { printf("\tReader is reading.\n"); }
void WRITEUNIT() { printf("\tWriter is writing.\n"); }

void writer() {
	printf("\n\nWriter:\n");
	if(!wait(&wsem)) {
		printf("\tWriter cannot write because reading is in progress.\n");
		return;
	}
	WRITEUNIT();
	signal(&wsem);
}

void reader(int w) {
	printf("\n\nReader:\n");
	if(!wait(&x)) { printf("\tReader cannot read because readcount is locked\n");  return; }
	readcount++;
	if(readcount==1)
		wait(&wsem);
	signal(&x);
	READUNIT();
	if(w==1) { writer(); signal(&wsem); }
	if(!wait(&x))  { printf("\tReader cannot quit because readcount is locked\n");  return; }
	readcount--;
	if(readcount==0)
		signal(&wsem);
	signal(&x);
}


void main() {

	x.count = 1;
	wsem.count = 1;

	clrscr();

	reader(0);

	reader(1);

	writer();

	getch();

}
```

OUTPUT

```
Reader:
        Reader is reading.

Reader:
        Reader is reading.

Writer:
        Writer cannot write because reading is in progress.

Writer:
        WRITER is writing.
```