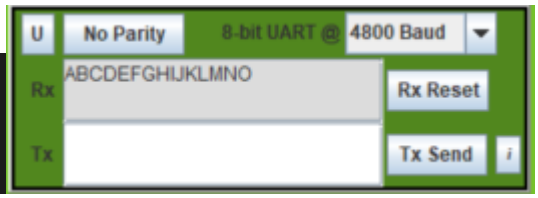
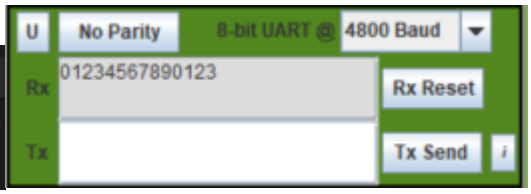
1. [40 points] `test3threads.c`

```
ThreadCreate(Producer1);
ThreadCreate(Producer2);
Consumer();
```




```
ThreadCreate(Producer2);
ThreadCreate(Producer1);
Consumer();
```

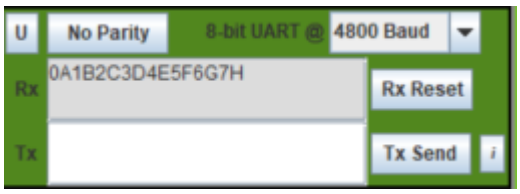


2. [40 points] Fairness

Originally, Thread 0 is consumer, 1 is producer1, 2 is producer2. Entering thread 1, buffer is ‘A’ and full is set to 1, so next thread (thread 2) can’t change anything. Then it goes to thread 0, and it writes down ‘A’, set full to 0, then it enter thread 1 and it goes on. So, thread 2 will starve forever.

```
void Producer1(void) {
        character = 'A';
        while (1) {
                SemaphoreWaitBody(turn1,  L(__COUNTER__));
                SemaphoreWaitBody(empty, L(__COUNTER__) );
                SemaphoreWaitBody(mutex, L(__COUNTER__) );
                buffer = character;
                if(character == 'Z') character = 'A';
                else character +=  1;
                SemaphoreSignal(mutex);
                SemaphoreSignal(full);
                SemaphoreSignal(turn2);
        }
}
void Producer2(void) {
        number = '0';
        while (1) {
                SemaphoreWaitBody(turn2,  L(__COUNTER__));
                SemaphoreWaitBody(empty, L(__COUNTER__) );
                SemaphoreWaitBody(mutex, L(__COUNTER__) );
                buffer = number;
                if(number == '9') number = '0';
                else number +=  1;
                SemaphoreSignal(mutex);
                SemaphoreSignal(full);
                SemaphoreSignal(turn1);
        }
}
```

By adding turn1 and turn2, producers 1 and 2 have to wait for the other's signal to be able to produce again. So, they take turn every time, which is fair.



3. [20 points] Typescript and screenshots

3.1 [2 points] Typescript for compilation

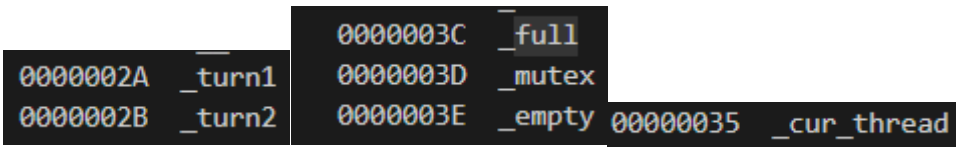
```
$ make clean
del *.hex *.ihx *.lnk *.lst *.map *.mem *.rel *.rst *.sym *.asm *.lk

$ make
sdcc -c    test3threads.c
test3threads.c:49: warning 158: overflow in implicit constant conversion
sdcc -c    preemptive.c
preemptive.c:152: warning 85: in function ThreadCreate unreferenced function
argument : 'fp'
preemptive.c:189: warning 158: overflow in implicit constant conversion
sdcc    -o test3threads.hex test3threads.rel preemptive.rel
```

3.2 [18 points] Screenshots and explanation

- Take screenshots when the `Producer1` and `Producer2` running and show semaphore changes.

  `Producer1` before:



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	30	30	00	00	00	00	00	01	31	00	00	00	00	00	00	02
10	30	31	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	01	00	00	00	00	00	00	00	00	00	01	00	00	00	00	00
30	46	56	66	00	07	01	00	41	02	00	00	41	00	01	01	00
40	97	00	00	00	02	00	90	30	00	00	00	00	00	00	00	00
50	14	00	00	00	00	00	09	00	00	00	00	00	00	00	00	00
60	4D	00	00	00	00	00	11	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

35 is 1, so now it is in producer1. And turn1(2A), mutex(3D), empty(3E) are set to 1, so producer1 is ready to set buffer to character.

`Producer1` after:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	30	30	00	00	00	00	00	01	31	00	00	00	00	00	00	41
10	30	31	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	01	00	00	00	00	00	00	00	00	00	00	01	00	00	00	00
30	46	56	66	00	07	01	00	41	02	00	41	42	01	01	00	00
40	97	00	00	00	02	00	90	30	00	00	00	00	00	00	00	00
50	14	00	00	00	00	00	09	00	00	00	00	00	00	00	00	00
60	4D	00	00	00	00	00	11	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

we can see that full(3C), turn2(2B) are set to 1, and turn1(2A), empty(3E) are 0, mutex(3D) decreased then increased so nothing changed.

`Producer2` before:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	30	30	00	00	00	00	00	01	31	32	00	00	00	00	00	41
10	30	31	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	01	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	46	56	66	00	07	02	00	41	02	00	41	42	00	01	01	30
40	95	00	00	00	02	00	90	30	30	00	00	00	00	00	00	00
50	17	00	00	00	00	00	08	31	32	00	00	00	00	00	41	00
60	59	00	00	00	00	00	10	32	31	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

35 is 2, so now it is in producer2. And mutex(3D), empty(3E) are set to 1, so producer2 is ready to set buffer to character. Notice that turn2(2B) is 0 since it is already waited in the last visit, but during last visit, producer1 had just finished, so empty is 0, meaning that producer2 have to wait for next time to produce.

`Producer2` after:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	30	30	00	00	00	00	00	01	31	32	00	00	00	00	00	41
10	30	31	00	00	00	00	00	30	00	00	00	00	00	00	00	00
20	01	00	00	00	00	00	00	00	00	00	01	00	00	00	00	00
30	46	56	66	00	07	02	00	41	02	00	30	42	01	01	00	31
40	95	00	00	00	02	00	90	30	30	00	00	00	00	00	00	00
50	17	00	00	00	00	00	08	31	32	00	00	00	00	00	41	00
60	59	00	00	00	00	00	10	32	31	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

we can see that full(3C), turn1(2A) are set to 1, and empty(3E) is 0, mutex(3D) decreased then increased so nothing changed.

- Take screenshots when the Consumer is running and show semaphore changes.

before:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	30	30	00	00	00	00	00	01	31	32	00	00	00	00	00	41
10	32	30	00	00	00	00	00	30	00	00	00	00	00	00	00	00
20	01	00	00	00	00	00	00	00	00	00	01	00	00	00	00	00
30	46	56	66	00	07	00	00	41	02	00	30	42	01	01	00	31
40	95	00	00	00	02	00	90	30	30	00	00	00	00	00	00	00
50	17	00	00	00	00	00	08	31	32	00	00	00	00	00	41	00
60	50	00	00	00	00	00	10	32	31	00	00	00	00	00	30	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

35 is 0, so now it is in Consumer. And full(3C) and mutex(3D) are set to 1, so it is ready to write to SBUF.

after:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	30	30	00	00	00	00	00	01	31	32	00	00	00	00	00	41
10	32	30	00	00	00	00	00	30	00	00	00	00	00	00	00	00
20	01	00	00	00	00	00	00	00	00	00	01	00	00	00	00	00
30	46	56	66	00	07	00	00	41	02	00	30	42	00	01	01	31
40	95	00	00	00	02	00	90	30	30	00	00	00	00	00	00	00
50	17	00	00	00	00	00	08	31	32	00	00	00	00	00	41	00
60	50	00	00	00	00	00	10	32	31	00	00	00	00	00	30	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

we can see that full(3C) is 0 and empty(3E) is set to 1, mutex(3D) decreased then increased so nothing changed.