1. [50 points] delay(*n*), now()
    1. My time unit is every 16th time that timer0_ISR is called. The period of the timer 0 ISR = (12 / (11.0592 * 10^6)) * 8192 = 8.8889 ms，so the time unit is around 0.14222seconds. By polling, continuously check if n time units have passed, and only exit the delay function when n time units have passed. This way, I can ensure that the delay is at least n time units but less than (n + 0.5)

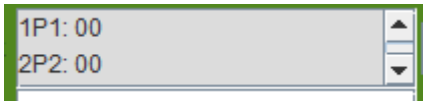
    2. Every time timer0 ISR occurs, increase the unit count by 1, when it reaches 16, increase the timer, set unit count to 0.

    3. After finish delaying, the threads take turn to do ThreadExit. By polling, I and ensure the accuracy is between n and n+0.5 time units.

    4. If all threads call delay() and happen to finish their delays at the same time, the accuracy of the delay can be affected, resulting in a delay that is longer than the specified time. There are total 4 threads, and the time unit increase every 16 context switch, so the worst case is when all threads finish delaying at the same time. thread A has to wait for 4 more context switch to finish, which is 1/4 of time units, so total delay is n+ 0.25 which is affected but didn't exceed.

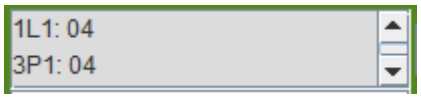
4. [20 points] Typescript and screenshots

```
$ make clean
del *.hex *.lst *.map *.mem *.rel *.rst *.sym *.asm *.lk
$ make
sdcc -c    testparking.c
In file included from testparking.c:2:
preemptive.h:31:18: warning: backslash and newline separated by space
testparking.c:104: warning 158: overflow in implicit constant conversion
sdcc -c    preemptive.c
In file included from preemptive.c:3:
preemptive.h:31:18: warning: backslash and newline separated by space
preemptive.c:124: warning 85: in function ThreadCreate unreferenced function
argument : 'fp'
sdcc    -o testparking.hex testparking.rel preemptive.rel
```



car1 park in space1 time: 00

car2 park in space2 time: 00



car1 leave space1 time: 04

car3 park in space1 time: 04

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	37	2B	00	00	00	00	00	00	2C	2D	00	00	00	00	00	34
10	2D	2E	00	00	00	00	00	0A	35	3B	00	00	00	00	00	04
20	35	01	00	34	32	33	00	01	00	33	32	46	56	66	76	0F
30	04	0A	00	04	0A	04	04	04	00	04	0A	08	03	03	41	01
40	D7	03	00	00	01	00	48	2B	2D	00	00	00	00	00	34	00
50	18	00	00	00	00	00	08	2C	2C	00	00	00	00	00	34	00
60	E1	01	0A	00	00	00	10	2D	3A	00	00	00	00	00	0A	00
70	E5	01	00	00	04	00	D8	2E	25	00	00	00	00	00	04	00

```
00000020  _car
00000021  _car_temp
00000022  _car_name
00000026  _empty
00000027  _mutex
00000028  _thread
00000029  _space1
0000002A  _space2
0000002B  _saved_sp
0000002F  _mask
00000030  _time
00000031  _time_unit
00000032  _time_car
00000038  _time_temp
0000003C  _cur_thread
0000003D  _i
0000003E  _temp
0000003F  _new_thread
```

0x30 is time which is total 04 time unit, 0x22, 0x24, 0x23, 0x25 is the car_name，where 0x22 is the main thread 0 so no car there, we can see car4 is in thread1, car2 in thread2, car 3 in thread3. 0x38, 0x39, 0x3A, 0x3B is the time_temp, which indicate when will the car leave the parking space, 0x38 is the main thread so 0, 0x39 indicate the last car in thread1 left in 04, car2 in thread2 will leave in 0A, car3 in thread3 will leave in 08. 0x33 ~ 0x37 store how long will a car stay, all will stay for 4 time units, only car2 will stay for 0A car units.