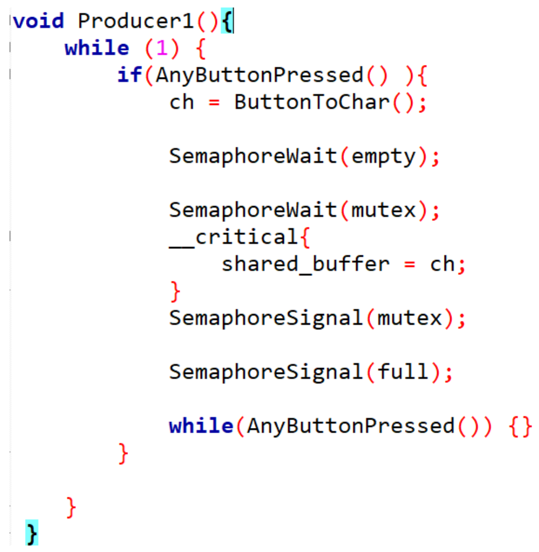
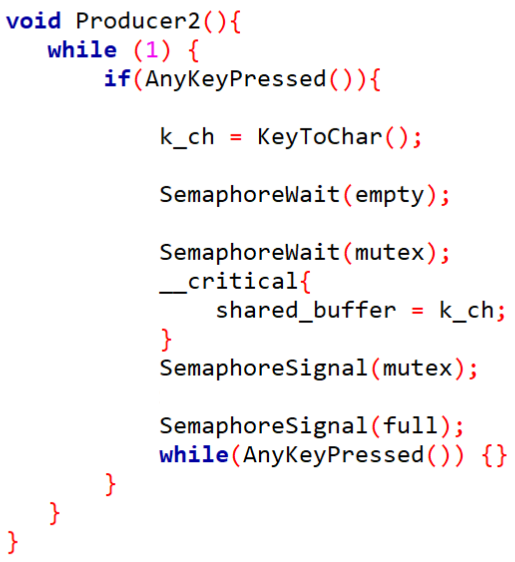
Programming Project Checkpoint 5

111065541 張騰午

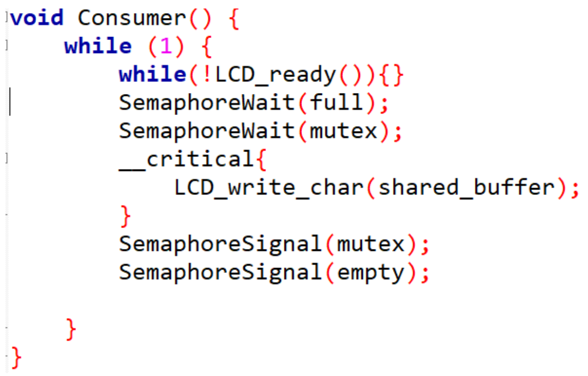
(Part 1)



```
void Producer1(){
    while (1) {
        if(AnyButtonPressed() ){
            ch = ButtonToChar();

            SemaphoreWait(empty);

            SemaphoreWait(mutex);
            __critical{
                shared_buffer = ch;
            }
            SemaphoreSignal(mutex);

            SemaphoreSignal(full);

            while(AnyButtonPressed()) {}
        }

    }
}
```



```
void Producer2(){
    while (1) {
        if(AnyKeyPressed()){

            k_ch = KeyToChar();

            SemaphoreWait(empty);

            SemaphoreWait(mutex);
            __critical{
                shared_buffer = k_ch;
            }
            SemaphoreSignal(mutex);

            SemaphoreSignal(full);
            while(AnyKeyPressed()) {}
        }
    }
}
```



```
void Consumer() {
    while (1) {
        while(!LCD_ready()){}
        SemaphoreWait(full);
        SemaphoreWait(mutex);
        __critical{
            LCD_write_char(shared_buffer);
        }
        SemaphoreSignal(mutex);
        SemaphoreSignal(empty);

    }
}
```

Three threads use shared_buffer to write character on LCD, use semaphore for race condition.

(Part 2)

Get difficulty level

After the initialization we first get the level from user, it should be 0~9 it means how many delay between two shift screen. If the input isn't a number level will be 0.

```c
while(1){
    if(AnyKeyPressed()){

        if(KeyToChar() == '#'){
            while(AnyKeyPressed()) {}
            //LCD_write_char('1');
            while(1){
                if(AnyKeyPressed()){
                    level = KeyToChar() - '0';
                    level = (level > 9) ? 0 : level;
                    while(AnyKeyPressed()) {}
                    break;
                }
            }
            break;

        }else{
            while(AnyKeyPressed()) {}
        }
    }
}
```

```c
void render_task(){
    int k;
    while (shared_buffer != 3) {
        SemaphoreWait(empty);
        SemaphoreWait(mutex);

        shift_screen1();

        SemaphoreSignal(mutex);
        SemaphoreSignal(full);
        k = level;
        delay(200);
        delay(200);
        delay(200);
        while(k--) { delay(200);}
        while(!LCD_ready()){}
    }
    end_scene();

    while(1){}

}
```

render_task thread

use shift_screen1() to update the screen and information of current state. Semaphore to protect this section. Do until game control use shared_buffer to end the game.

keypad_ctrl thread

Get key pressed signal and judge the movement of dino. Also use semaphore to protect shared_buffer. It contains the dino position.

```c
void keypad_ctrl(){
    while (1) {
        if(AnyKeyPressed()){
            k_ch = KeyToChar();
            if(k_ch == '8' & shared_buffer == 0){
                SemaphoreWait(empty);
                SemaphoreWait(mutex);
                while(!LCD_ready()){}
                LCD_cursorGoTo(shared_buffer, 0);

                shared_buffer = 1;

                LCD_write_char(' ');
                while(!LCD_ready()){}
                LCD_cursorGoTo(shared_buffer, 0);
                LCD_write_char('\2');

                SemaphoreSignal(mutex);
                SemaphoreSignal(full);
                while(AnyKeyPressed()) {}
            }else if(k_ch == '2' & shared_buffer == 1){
                SemaphoreWait(empty);
                SemaphoreWait(mutex);
                while(!LCD_ready()){}
                LCD_cursorGoTo(shared_buffer, 0);

                shared_buffer = 0;

                LCD_write_char(' ');
                while(!LCD_ready()){}
                LCD_cursorGoTo(shared_buffer, 0);
                LCD_write_char('\2');
                SemaphoreSignal(mutex);
                SemaphoreSignal(full);
                while(AnyKeyPressed()) {}
            }else{
                while(AnyKeyPressed()) {}
            }
        }
    }
}
```

```
void game_control() {

    while (1) {
        SemaphoreWait(full);
        SemaphoreWait(mutex);
        //判斷state
        if(at0 == shared_buffer){
            shared_buffer = 3;
        }
        SemaphoreSignal(mutex);
        SemaphoreSignal(empty);


    }
}
```

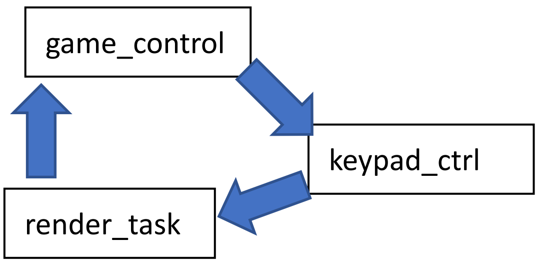
game_control thread

This thread check the value of at0 and share_buffer to judge the game. I use share_buffer to store the dino position, 0 means (0,0) and 1 means (1,0). at0 store if there's a cactus in position (0,0) or (1,0).

How the three threads work together?

In preemptive.c, I do round-robin policy for threads. In dino.c the first thread(main func) do render_task, the second thread is game_control and the third one is keypad_ctrl. The three threads will be execute by the order 1,2,3,1,2,3,1……If the first thread, render_task, acquires the semaphore, game_control judges and signals the semaphore afterward. Following the SemaphoreSignal, keypad_ctrl executes immediately afterward. This implies that if we press a key to move the dino, keypad_ctrl acquires the semaphore with priority, ensuring that the press can be executed without concern.

```
i = current_thread_id;
do {
    i++;
    if (i >= MAXTHREADS) {
    i = 0;
    }

    if ((threads_available >> i) & 1) {
    break;
    }
} while (1);
```



Need sema?
You choose
first then
next guy

Questions

What data type do you use for the map?

I have 5 cactuses 2 in row 0, and three in row 1. I use two char arrays to store these two kinds of cactuses position(0~14).

```
__idata __at (0x3a) char pos_c0[2];//0x3a~0x3b
__idata __at (0x3c) char pos_c1[3];//0x3c~0x3e
```

Each time shifting the screen the render_task thread also update the new position information of each cactus.

When there's a cactus in position (0,0) or (1,0), I use another char variable at0 to the information. at0 = 0 for there's a cactus in (0,0), at0 = 1 for in (1,0), otherwise, at0 =2.

```
__idata __at (0x34) char at0;
```

How do you generate a new cactus?

Simple policy, if current position is (0,0)/(1,0), the next position is (0,14)/(1,14). That's a "new" cactus.

```
if(pos_c0[i]!=0){pos_c0[i]--;}
else {pos_c0[i] = 14; score++;}
```

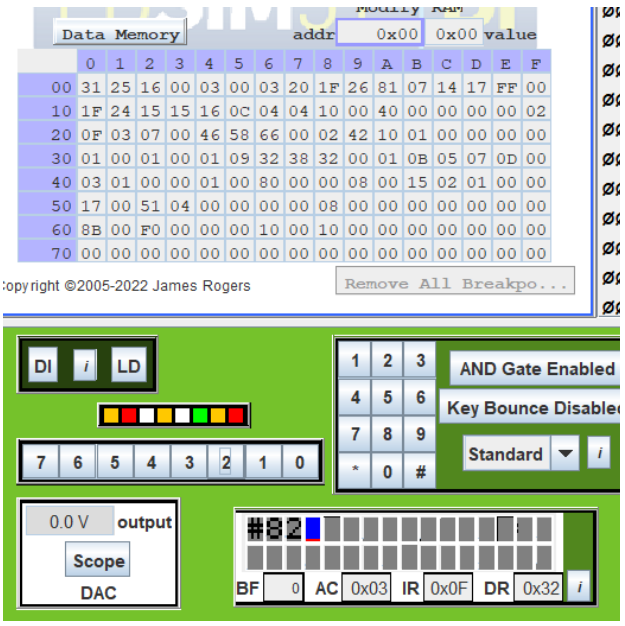
If the initial cactuses is fine the new one will be, too.

What variables may race conditions and why?

shared_buffer and at0. I use share_buffer to store the dino position, 0 means (0,0) and 1 means (1,0). at0 store if there's a cactus in position (0,0) or (1,0). We update the at0 value in render_task, and update the share_buffer value in keypad_ctrl. In game_control thread, we check the value of at0 and share_buffer to judge the game. So we use the semaphore.

```
SemaphoreWait(full);
SemaphoreWait(mutex);
//判斷state
if(at0 == shared_buffer){
    shared_buffer = 3;
}
SemaphoreSignal(mutex);
SemaphoreSignal(empty);
```

Typescript and screenshots


Data Memory addr 0x00 0x00 value

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	31	25	16	00	03	00	03	20	1F	26	81	07	14	17	FF	00
10	1F	24	15	15	16	0C	04	04	10	00	40	00	00	00	00	02
20	0F	03	07	00	46	58	66	00	02	42	10	01	00	00	00	00
30	01	00	01	00	01	09	32	38	32	00	01	0B	05	07	0D	00
40	03	01	00	00	01	00	80	00	00	08	00	15	02	01	00	00
50	17	00	51	04	00	00	00	00	08	00	00	00	00	00	00	00
60	8B	00	F0	00	00	00	10	00	10	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

:opyright ©2005-2022 James Rogers

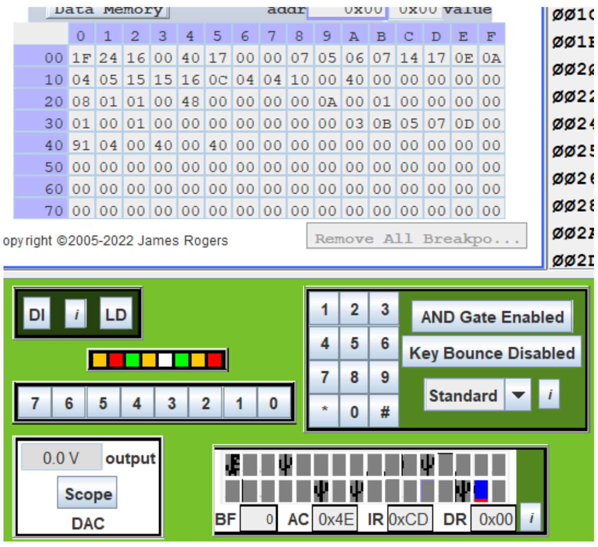
AND Gate Enabled

Key Bounce Disable

Standard

0.0 V output

BF 0 AC 0x03 IR 0x0F DR 0x32

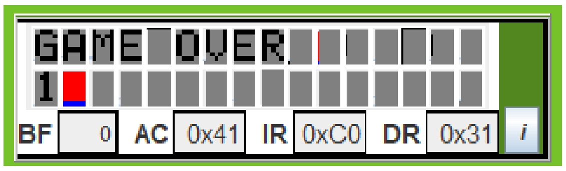


```
    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00 1F 24 16 00 40 17 00 00 07 05 06 07 14 17 0E 0A
10 04 05 15 15 16 0C 04 04 10 00 40 00 00 00 00 00
20 08 01 01 00 48 00 00 00 00 0A 00 01 00 00 00 00
30 01 00 01 00 00 00 00 00 00 00 03 0B 05 07 0D 00
40 91 04 00 40 00 40 00 00 00 00 00 00 00 00 00 00
50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
60 00 00 00 00 00 00 00 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
```

opyright ©2005-2022 James Rogers

0.0 V output

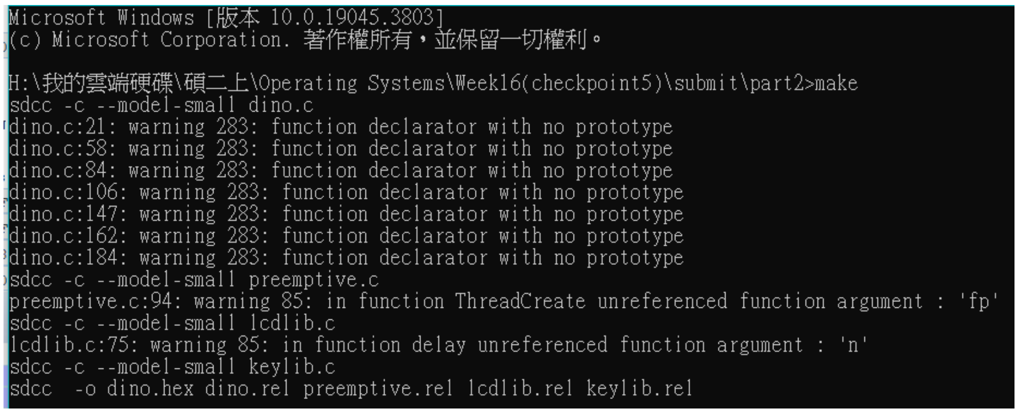
BF 0 AC 0x4E IR 0xCD DR 0x00



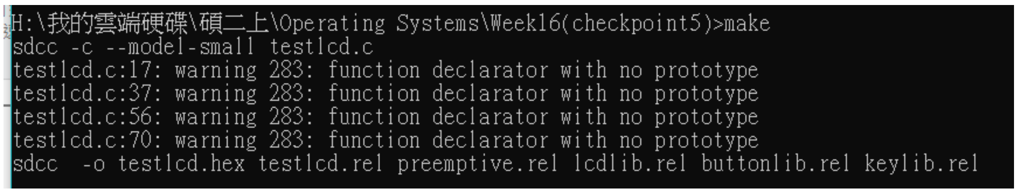
```
GAME OVER
1
```

BF 0 AC 0x41 IR 0xC0 DR 0x31

compilation screen :



```
Microsoft Windows [版本 10.0.19045.3803]
(c) Microsoft Corporation. 著作權所有，並保留一切權利。

H:\我的雲端硬碟\碩二上\Operating Systems\Week16(checkpoint5)\submit\part2>make
sdcc -c --model-small dino.c
dino.c:21: warning 283: function declarator with no prototype
dino.c:58: warning 283: function declarator with no prototype
dino.c:84: warning 283: function declarator with no prototype
dino.c:106: warning 283: function declarator with no prototype
dino.c:147: warning 283: function declarator with no prototype
dino.c:162: warning 283: function declarator with no prototype
dino.c:184: warning 283: function declarator with no prototype
sdcc -c --model-small preemptive.c
preemptive.c:94: warning 85: in function ThreadCreate unreferenced function argument : 'fp'
sdcc -c --model-small lcdlib.c
lcdlib.c:75: warning 85: in function delay unreferenced function argument : 'n'
sdcc -c --model-small keylib.c
sdcc  -o dino.hex dino.rel preemptive.rel lcdlib.rel keylib.rel
```



```
H:\我的雲端硬碟\碩二上\Operating Systems\Week16(checkpoint5)>make
sdcc -c --model-small testlcd.c
testlcd.c:17: warning 283: function declarator with no prototype
testlcd.c:37: warning 283: function declarator with no prototype
testlcd.c:56: warning 283: function declarator with no prototype
testlcd.c:70: warning 283: function declarator with no prototype
sdcc  -o testlcd.hex testlcd.rel preemptive.rel lcdlib.rel buttonlib.rel keylib.rel
```