CSE2037 Kendimce Çözümler

1. Vizedeki Sorunun Gelişmişi

```
#include "mbed.h"

#define WAIT_MSEC 20PwnOut LED(p20);

InterruptIn enter_Sensor(p10);
InterruptIn enter_Sensor(p10);

int totalNumberofCustomers = 0;
float brightness = 0;
// This ISR will be add 1 to
// the totalNumberofCustomers.

void enter_ISR() {
    extern int totalNumberofCustomers;
    totalNumberofCustomers += 1;
}

// This ISR will be suvsbtroct 1 from
// the totalNumberofCustomers.
void exit_ISR() {
    extern int totalNumberofCustomers;
    totalNumberofCustomers -= 1;
}

int main() {
    enter_Sensor.fall(callback(&enter_ISR));
    exit_Sensor.fall(callback(&enter_ISR));

    while(true)
        extern int totalNumberofCustomers;
        extern float brightness;
        if !(totalNumberofCustomers) brightness = 0;
        else {
            // Because we want 20 people for
            // most bright state, the step
            // should be 1/20 = 0.05
            // This will be the incremant for
            // duty cycle at every person enters.
            // LED is a PwnOut pin.
            brightness = totalNumberofCustomers * 0.05f;
        }
        LED = brightness;
        wait_ms(WAIT_MSEC);
}
```

2. Riemman İntegrali Alma

```
#include "mbed.h"

#define 5MIN_SEC 300

AnalogIn source(p20);

Timeout tout;

volatile int cond = 1;
int total = 0;

void timeout_ISR() {
    extern volatile int cond;
    cond = 0;
}

int main() {
    extern volatile int cond;
    tout.attach(callback(&timeout_ISR), 5MIN_SEC);

    while (cond) {
        total += source.read();
        wait_ms(20); // 0.02sec = 20ms
    }

    wait_ms(osWaitForever);
}
```

3. Bit Değişim Sorusu

```
#include <iostream>

int main() {
  uint16_t x = 0b1101111011101111;
  uint16_t mask = 0b0000000011111111;
  uint16_t newV = x & mask;
  uint8_t x_low = newV;

  uint8_t x_high = x >> 8;

  uint16_t x_swap = x_low;
  x_swap = x_swap << 8;
  x_swap += x_high;

  return 0;
}
```

4. Assembly Sorusu

```
sum_ADDR equ     0x200020

         LDR     R0, =sum_ADDR
         MOV     R1, #0
         STR     R1, [R0]
```

```
6
7           LDR     R2, =0x20000010
8           LDR     R3, =0x20000014
9           LDR     R4, =0x20000018
10
11          LDR     R2, [R2]
12          LDR     R3, [R3]
13          LDR     R4, [R4]
14
15          MOV     R5, #2
16  adding
17          ADD     R1, R1, R2
18          ADD     R1, R1, R3
19          ADD     R1, R1, R4
20          SUBS    R5, R5, #1
21          BPL     adding
```