Palindrome
How many carry operations?
How many ones?

전체적인 평

- 좋음.

Palindrome를 찾아서

```c
#include <stdio.h>
void reverse()
{
    int num;
    if((num = getchar()) != '₩n')
    {
        reverse();
        putchar(num);
    }
}

void sum()
{
    int num, sum, palindrome;
    sum = num + palindrome;
}
```

```c
int _main()
{
    int num;
    scanf("%d", &num);

    if(num<0)
        printf("Overflow₩n");
    else
    {
        while(num > 0)
        {
            printf("%d", num % 10);
            num = num / 10;
        }
    }

    printf("₩n");

    return 0;
}
```

문제를 이해하지 못한 코드

```
#define BOOL int
#define TRUE 1
#define FALSE 0


BOOL isPalindrome(int n) {
    int r;

    r = calculateReverse(n);

    if((n - r) == 0)
        return TRUE;

    return FALSE;
}
```

좋은 습관!!

```
int _main(void)
{
    int input, count = 1, sum = 0;

    scanf("%d", &input);

    sum = input + calculateReverse(input);

    while(sum > 0 | isPalindrome(sum) == TRUE) {
        count++;
        sum = sum + calculateReverse(sum);

        if(isPalindrome(sum) == TRUE) {
            printf("%d %d₩n", count, sum);
            break;
        }
    }

    if(sum < 0)
        printf("Overflow₩n");
}
```

Indent, 줄바꿈 모두 좋은 습관!!

입력한 숫자가 Palindrome이면?

```
int _main()
{
    long long num;
    int cnt = 0;
    int reverse;
    scanf( "%lld" , &num);
    while (1)
    {
        reverse = calculateReverse(num);
        if (reverse == 0)
            break;

        cnt++;
        num += reverse;

    }

    if (num < 0)
        printf("Overflow");
    else
        printf("%d %lld", cnt, num);
}
```

Loop 수행 중에 num 변수가 Overflow 되면?

if (num<0 || num<reverse) break;

How many carry operations?

```
int _main (_void)
{
    int carryCount = 0;
    int n, m;;

    scanf("%d %d", &n, &m);

    while(n != 0 && m != 0)
    {
        if(n % 10 + m % 10 >= 10)
            carryCount += 1;

        n = n %10;
        m = m % 10;
    }

    printf("%d₩n", carryCount);
}
```

문제를 이해하지 못한 코드

```
int calculateCarry(int n, int m)
{
    int n1, m1;
    int carry = 0;

    while (n != 0 && m != 0)
    {
        n1 = n % 10;
        m1 = m % 10;

        if (n1 + m1 >= 10) {
            carry++;
            n = n / 10 + 1; // carry 처리하는 것이 오류
            m /= 10;
        }
        else {
            n /= 10;
            m /= 10;
        }

    }
    return carry;
}
```

99+10이면?
두 번 carry가 발생
이 코드는 1회만 발생

```
int _main(void){

    int n1, n2;
    int count = 0, temp;
    int t = 10;

    scanf("%d %d", &n1, &n2);

    while(t <= n1 + n2){
        temp = n1 % 10 + n2 % 10;
        if(temp >= t)
            count++;
        t *= 10;
    }
    printf("%d₩n", count);
}
```

오른쪽으로 shift해야함

```
int _main(){
    int x = 0, y = 0;
    int carry = 0;

    scanf("%d %d", &x, &y);
    while(x+y > 9){
        if((x%10 + y%10) >= 10){
            carry++;
        }
        x /= 10;
        y /= 10;
    }
    printf("%d₩n", carry);
}
```

특이한 조건 (99+1이면 10의 자리수 연산 안함)

이전 자리에서 carry 발생하면 1을 더해야

오른쪽으로 shift해야함

How many ones?

문제를 이해하지 못한 코드가 여전히…

```
int calculateReverse(int n)
{
    int a;

    a = 0;
    while (n > 0)
    {
        a *= 10;

        a += (n % 10);

        n /= 10;
    }

    return a;
}
```

```
int _main(void)
{
    int num, result;
    int i = 0;
    int count = 0;
    scanf("%d", &num);
    do
    {
        i++;
        result = num * i;
    } while (result != calculateReverse(result) || result % 10 != 1);
    while (result > 0)
    {
        count++;
        result /= 10;
    }
    printf("%d₩n", count);
}
```

숙제 미제출과 동치!!

```
#include <stdio.h>
int _main(void){
    int i, n;
    int b=0, c = 0;

    scanf("%d", &n);

    for(i = 0; i < n; i++){

    }
}
```

```
int findOne(int n)
{
    int cnt = 0;
    int num[100];
    int i;


    while (n > 0)
    {
        num[cnt] = n % 10;
        cnt++;
        n /= 10;
    }

    for (i = 0; i < cnt; i++)
    {
        if (num[i] != 1)
            return 0;
    }
    return cnt;
}
```

```
int main(void)
{
    int n, m;
    int k = 1;

    scanf("%d", &n);

    while (1)
    {
        m = n * k;
        findOne(m);

        if (findOne(m) != 0)
            break;
        k++;
    }

    printf("%d₩n", findOne(m));
}
```

왜 같은 인자로 3번 호출?

결과는 맞지만
시간 초과

```
int find(int num)
{
    int n = num, count = 0, i, remain, sum = 0;

    do {
        n /= 10;
        count++;
    } while (n > 0);

    for (i = 0; i < count; i++)
    {
        remain = num % 10;
        num /= 10;
        sum += remain;
    }
    if (sum == 1 * count)
        return count;
}
```

왜 같은 인자로 2번 호출?

```
int _main(void)
{
    int a, b = 1, c;

    scanf("%d", &a);

    do {
        c = a * b;
        if (!find(c)) {
            b++;
            continue;
        }
    } while (!find(c));

    printf("%d₩n", find(c));
}
```

1. return0이 안 되는 경우도 있다.
2. 코드가 4자리의 수가 각 자리의 수가 40이면 참으로 리턴. ^^
   안 되는 경우도 있음. num이 10210이면 참임.

Bit로 문제를 확장하면?

int a = 9; // 00000000000000000000000000001001
int b = ??; // 11000000000000000000000000000011

a가 bit 수준에서 palindrome인지? (ans : 아님)
b는 bit 수준에서 palindrome임.

다시 이전 과제로 돌아가면.. (십진수)

int a = 11721;

a가 palindrome인지 확인하기 위해
a의 reverse를 만들었음.

int b = 0;

b = a%10; // b ← 1 (a의 마지막 자리수)
a = a/10; // a ← 1172 (a를 오른쪽으로 shift)
b = b*10 + a%10; // b ← 12 (b를 왼쪽으로 shift하고 a의 마지막 자리수 더함)
a = a/10;
b = b*10 + a%10; // b ← 127
…

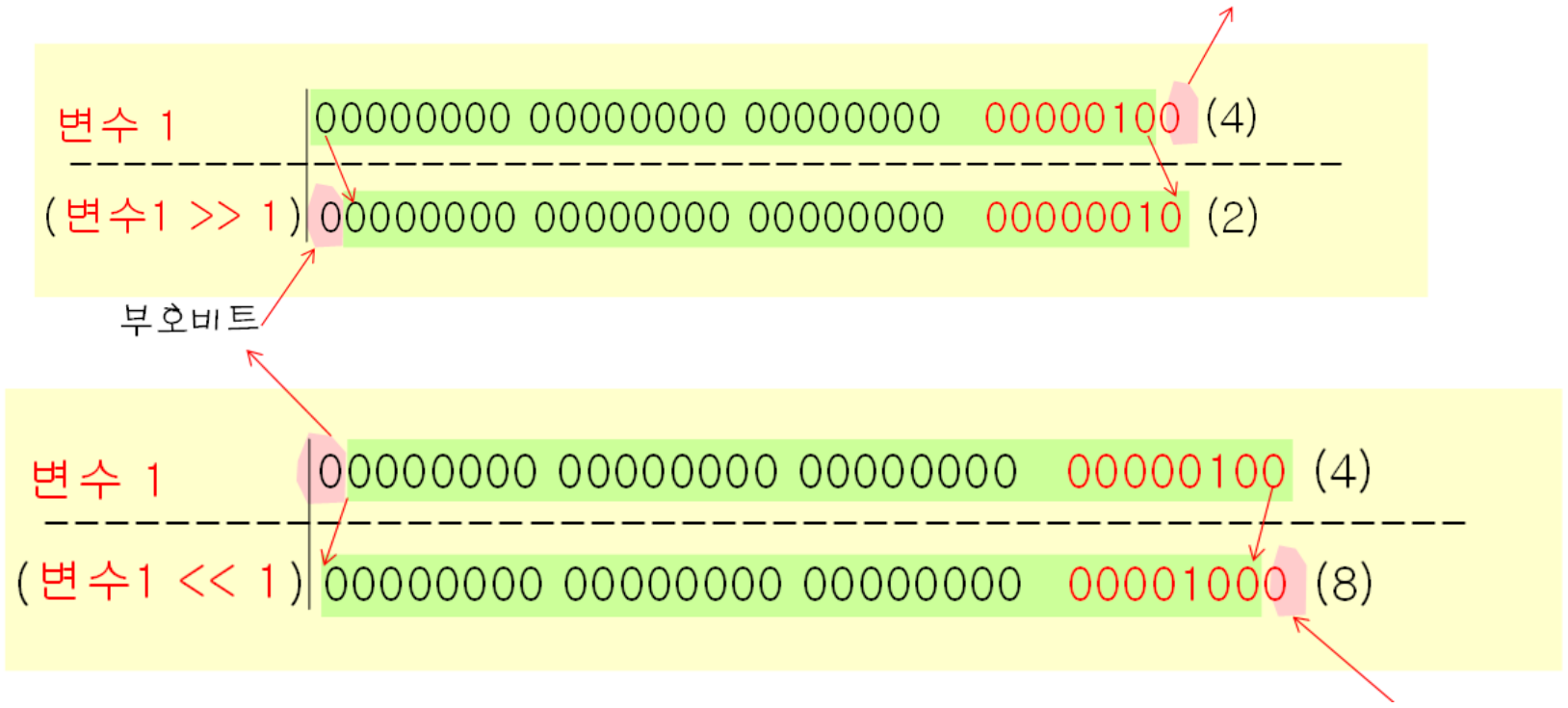
2진수에서도 비슷하다.
가장 쉬운 방법은 10을 2로 치환
Key idea는
1) 마지막 자리 수를 알아내는 방법 (a%2)
2) 오른쪽으로 shift (a/2)
3) 왼쪽으로 shift (b*2)
이 세 가지 연산으로 위의 숙제를 모두 비트 수준(2진수)에서 모두 구할 수 있다.

```
int a = 11721;
int b = 0;

b = a%2; // b ← a의 마지막 비트
a = a/2; // a를 오른쪽으로 비트 수준으로 shift
b = b*2 + a%2; // b ← b를 왼쪽으로 비트 수준으로 shift하고 a의 마지막 비트 더함
a = a/2;
b = b*2 + a%2;
…
```

2진수에서도 비슷하다.
/,%,* 대신에 bit-wise operator를 사용할 수도 있다. (사실 이 방법이 더 선호됨)
Key idea는
1) 마지막 자리 수를 알아내는 방법 (a&1)
2) 오른쪽으로 shift (a>>1)
3) 왼쪽으로 shift (b<<1)
이 세 가지 연산으로 위의 숙제를 모두 비트 수준(2진수)에서 모두 구할 수 있다.

```
int a = 11721;
int b = 0;

b = a&1; // b ← a의 마지막 비트
a = a>>1; // a를 오른쪽으로 비트 수준으로 shift
…
```

2진수에서도 비슷하다.
/,%,* 대신에 bit-wise operator를 사용할 수도 있다. (사실 이 방법이 더 선호됨)
Key idea는
1) 마지막 자리 수를 알아내는 방법 (a&1)
2) 오른쪽으로 shift (a>>1)
3) 왼쪽으로 shift (b<<1)
이 세 가지 연산으로 위의 숙제를 모두 비트 수준(2진수)에서 모두 구할 수 있다.



2진수에서도 비슷하다.
/,%,* 대신에 bit-wise operator를 사용할 수도 있다. (사실 이 방법이 더 선호됨)
Key idea는
1) 마지막 자리 수를 알아내는 방법 (a&1)
2) 오른쪽으로 shift (a>>1)
3) 왼쪽으로 shift (b<<1)
이 세 가지 연산으로 위의 숙제를 모두 비트 수준(2진수)에서 모두 구할 수 있다.

변수 1 : 00000000000000000000000000001001 (9)
상수 1 : 00000000000000000000000000000001 (1)

(변수 1 & 상수 1) : 00000000000000000000000000000001 (1)

2진수에서도 비슷하다.
Key idea는
1) 마지막 자리 수를 알아내는 방법 (a&1)
2) 오른쪽으로 shift (a>>1)
3) 왼쪽으로 shift (b<<1)
이 세 가지 연산으로 위의 숙제를 모두 비트 수준(2진수)에서 모두 구할 수 있다.

int a = 11721;

a가 palindrome인지 확인하기 위해
a의 reverse를 만들었음.

int b = 0;

b = a&1; // b ← a의 마지막 비트
a = a>>1; // a를 오른쪽으로 비트 수준으로 shift
…