exercise

Bit manipulation ... Lecture 15

these and their variants

ex. a&=b

- `~` complement → flip bits $0 \leftrightarrow 1$
- `<<` left shift : always shifts in 0s
- `>>` right shift : shifts in 0s for unsigned types
- `&` and
- `|` or
- `^` xor

Typical questions : COMPLEMENT

- Assume 16-bit unsigned short :

  unsigned short n = 0xA2B3;

  Find ~n in hexadecimal.

$$n = \underbrace{0}_{\text{don't count this}}xA2B3 = \overset{A}{1010}\ \overset{2}{0010}\ \overset{B}{1011}\ \overset{3}{0011}$$

flip bits ↓

$$\sim n = \underbrace{\ }_{\text{don't count this}} = 0101\ 1101\ 0100\ 1100 = 0x5D4C$$

2510 : Lecture 15 || February 15, 2017

Note: C Review Exercise 2 NO 2. d), e)

TYPICAL QUESTIONS

- **Left shift** (<<)

  ex. unsigned short n = ØxA2B3;

  = 1010 0010 1011 0011 (000 ↓)

  n << 3 = 0001 0101 1001 1000 = Øx1598

doesn't usually ask for RSHIFT

- **Right shift** (>>) → shifts in Øs for unsigned types
  → " for non-neg. values in signed types for neg. values (signed types) which may shift in either Øs or 1s

  unsigned short n = ØxA2B3;

  = 1010 0010 1011 0011 (↑ 000)

  n >> 3 = 0001 0100 0101 0110 = Øx1456

- **And** (&) → 1 if both bits are 1; otherwise Ø

Find m & n: unsigned short m = Øx9C3F, n = ØxA2B3;

m = Øx9C3F = 1001 1100 0011 1111

n = ØxA2B3 = 1010 0010 1011 0011

m & n = 1000 0000 0011 0011 = Øx8033

▷ Typical Questions cont.

- OR (|) → ∅ if both bits are ∅; otherwise 1

Find m|n:

m = ∅x9C3F 1001 1100 0011 1111
n = ∅xA2B3 1010 0010 1011 0011

m|n = 1011 1110 1011 1111 = ∅xBEBF
m^n = 0011 1110 1000 1100 = ∅x3E8C

- XOR (^) → 1 if bits are different; otherwise ∅.

▷ Control Flow

1) if / else if / else

```
ex. if (argc == 1) {
        ...
    } else if (argc == 2) {
        ...
    } else {
        ...
    }
```

2) switch / case

```
ex. char choice = get_choice();
    switch (choice) {
      case 'y': case 'Y':
        ...
        break;
      case 'n': case 'N':
        ...
        break;
      default:
        ...
    }
```

must be a integral type — what?

Ask yourself: ① how many times is the loop run?
② what is printed the first time?

3) while (4 types)

```
int n = 1;
while (n++ < 5)
  printf("%d", n);
```

4x → 2 3 4 5

```
int n = 1;
while (++n < 5)
  printf("%d", n);
```

3x → 2 3 4

```
int n = 1;
while (n < 5)
  printf("%d", ++n);
```

4x → 2 3 4 5

```
int n = 1;
while (n < 5)
  printf("%d", n++);
```

4x → 1, 2 3 4

Know difference between n++ and ++n

→ ++n vs. n++

- ++n: value AFTER incrementing
- n++: value BEFORE incrementing

4) Do while

```
do {
...          /*exe. at least once*/
} while (...);
```

5) For Loop

```
for (initialization ; test to Continue ; update) {
  body
}
```

ex.
```
for (i = 0; i < 10 ; i++) {
  printf ("%d", i*i);
}
```

vs.

While loop
```
i = 0;
while (i < 10) {
  printf ("%d", i*i);
i++;
}
```

6) break / continue : must be used in loops

```
while (...) {
...
if (...)
  continue;
...
if (...)
  break;
...
}
```

ex.
```
while (i<10){
...
if (...)
  continue;
...
i++;
}
```

i is NOT incremented

vs.

```
for (...;...;i++){
  if (...)
    continue;
...
}
```

Standard Idiom to Process an array backwards

type — some positive integer

```
T a[N];
size_t i;
for (i = N; i > 0; i--) {
    /* process a[i-1] */
```

→ example to check if two strings are palindromes

```
int is_palindrome (const char []){
  size_t i;
  for (i = 0, j = strlen(s); i<j; i++, j--){
    if (s[i] != s[j-1]){
      return 0;
    }
  return 1;
}
```