CS61C — C & Number Representation

Summer 2025 — Discussion 1

1 Precheck: Number Representation

1.1 Depending on the context, the same sequence of bits may represent different things.

```
True
```

1.2 It is possible to get an overflow error in Two's Complement when adding numbers of opposite signs.

```
False
```

1.3 If you interpret a N bit Two's complement number as an unsigned number, negative numbers would be smaller than positive numbers.

```
False
```

1.4 If you interpret an N bit Bias notation number as an unsigned number (assume there are negative numbers for the given bias), negative numbers would be smaller than positive numbers.

```
True
```

1.5 We can represent fractions and decimals in our given number representation formats (unsigned, biased, and Two's Complement).

```
False
```

2 Unsigned and Signed Integers

2.1 Convert the following numbers from their initial radix into the other two common radices:

(a) `0b10010011`
十进制：147
十六进制：0x93

(b) `0`
二进制：0b0
十六进制：0x0

(c) `437`
二进制：0b110110101
十六进制：0x1B5

(d) `0x0123`
十进制：291
二进制：0b100100011

2.2 Convert the following numbers from hex to binary:

(a) `0xD3AD`

`0b1101001110101101`

(b) `0x7EC4`

`0b0111111011000100`

2.3 Assuming an 8-bit integer and a bias of −127 where applicable, what is the largest integer for each of the following representations? What is the result of adding one to that number?

(a) Unsigned
255

(b) Biased
-127 到128

(c) Two's Complement
-128到127

$\boxed{2.4}$ How would you represent the numbers 0, 1, and −1? Express your answer in binary and a bias of −127 where applicable.

(a) Unsigned

0, 1，无法表示

二进制：0000 0000,0000 0001

(b) Biased

127, 128, 126

二进制：0111 1111,1000 0000,0111 1110

(c) Two's Complement

二进制：0000 0000, 0000 0001, 1111 1111

$\boxed{2.5}$ How would you represent the numbers 17 and −17? Express your answer in binary and a bias of −127 where applicable.

(a) Unsigned

17, 无法表示

二进制：0001 0001

(b) Biased

144, 110

二进制：1001 0000，0110 1110

(c) Two's Complement

二进制：0001 0001，1110 1111

$\boxed{2.6}$ What is the largest integer that can be represented by *any* encoding scheme that only uses 8 bits?

255

$\boxed{2.7}$ Prove that that $x + \overline{x} + 1 = 0$, where $\overline{x}$ is obtained by inverting the bits of x in binary.

x+x取反 = 0b1111..111，故再加1会溢出一位其余为0，溢出位舍弃后结果为0

3 Arithmetic and Counting

$\boxed{3.1}$ Compute the decimal result of the following arithmetic expressions involving 6-bit Two's Complement numbers as they would be calculated on a computer. Do any of these result in an overflow? Are all these operations possible?

(a) `0b011001 - 0b000111`

`010010`

(b) `0b100011 + 0b111010`

`011101`，为正数，溢出

(c) `0x3B + 0x06`

转为二进制，得 000001

(d) `0xFF - 0xAA`

转为二进制后为八位不是六位

(e) `0b000100 - 0b001000`

`0b111100`

$\boxed{3.2}$ How many distinct numbers can the following schemes represent? How many distinct positive numbers?

(a) 10-bit unsigned

1024, 1023

(b) 8-bit Two's Complement

256, 127

(c) 6-bit biased, with a bias of −30

64, 33

(d) 10-bit sign-magnitude

1023, 511

4 Precheck: Introduction to C

4.1 The correct way of declaring a character array is `char[] array`.

False

4.2 True or False: C is a pass-by-value language.

True

4.3 In compiled languages, the compile time is generally pretty fast, however the run-time is significantly slower than interpreted languages.

False

4.4 What is a pointer? What does it have in common with an array variable?

指针存储地址，指向该地址的位置，数组本身也是指向首元素的地址

4.5 If you try to dereference a variable that is not a pointer, what will happen? What about when you free one?

解引用非指针类型变量会将该变量的值作为地址去访问，可能会出现非法地址的错误。释放未被分配堆内存的变量无效。

4.6 Memory sectors are defined by the hardware, and cannot be altered.

False

5 Pass-by-Who?

5.1 The following functions may contain logic or syntax errors. Find and correct them.

(a) Returns the sum of all the elements in `summands`.

```
int sum(int *summands) {
    int sum = 0;
    for (int i = 0; i < sizeof(summands) / sizeof(int); i++)
        sum += *(summands + i);
    return sum;
}
```

(b) Increments all of the letters in the `string` which is stored at the front of an array of arbitrary length, `n >= strlen(string)`. Does not modify any other parts of the array's memory.

```
void increment(char *string, int n) {
    for (int i = 0; string[i] != 0; i++)
        (*(string + i))++;
}
```

(c) Overwrites an input string `src` with “61C is awesome!” if there’s room. Does nothing if there is not. Assume that `length` correctly represents the length of `src`.

```
void cs61c(char *src, size_t length) {
    char *srcptr, *replaceptr;
    char replacement[16] = "61C is awesome!";
    srcptr = src;
    replaceptr = replacement;
    if (length >= 16) {
        for (int i = 0; i < 16; i++)
            *srcptr++ = *replaceptr++;
    }
}
```

5.2 Implement the following functions so that they work as described.

(a) Swap the value of two `int`s. *Remain swapped after returning from this function.* Hint: Our answer is around three lines long.

```
void swap(int *a, int *b) {
     int tmp = *a;
     *a = *b;
     *b = tmp;
}
```

(b) Return the number of bytes in a string. *Do not use `strlen`.* Hint: Our answer is around 5 lines long.

```
int mystrlen(char str[]) {
       int cnt = 0;
       while (str[cnt] != '\0') {
          cnt ++;
       }
       return cnt * sizeof(char);
}
```