

1 Precheck: Number Representation

- 1.1

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

- 1.2

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

- 1.3

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

- 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.

- 1.5

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

2 Unsigned and Signed Integers

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

(a) **0b10010011**

(b) **0**

(c) **437**

(d) **0x0123**

2.2 Convert the following numbers from hex to binary:

(a) **0xD3AD**

(b) **0x7EC4**

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

(b) Biased

(c) Two's Complement

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

(b) Biased

(c) Two's Complement

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

(b) Biased

(c) Two's Complement

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

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

3 Arithmetic and Counting

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) $0b011001 - 0b000111$

(b) $0b100011 + 0b111010$

(c) $0x3B + 0x06$

(d) $0xFF - 0xAA$

(e) $0b000100 - 0b001000$

3.2 How many distinct numbers can the following schemes represent? How many distinct positive numbers?

(a) 10-bit unsigned

(b) 8-bit Two's Complement

(c) 6-bit biased, with a bias of -30

(d) 10-bit sign-magnitude

4 Precheck: Introduction to C

- 4.1 The correct way of declaring a character array is `char[] array`.
- 4.2 True or False: C is a pass-by-value language.
- 4.3 In compiled languages, the compile time is generally pretty fast, however the run-time is significantly slower than interpreted languages.
- 4.4 What is a pointer? What does it have in common with an array variable?
- 4.5 If you try to dereference a variable that is not a pointer, what will happen? What about when you free one?
- 4.6 Memory sectors are defined by the hardware, and cannot be altered.

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); 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; i < n; 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 `ints`. *Remain swapped after returning from this function.* Hint: Our answer is around three lines long.

```
void swap(_____, _____) {
```

```
}
```

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

```
int mystrlen(_____) {
```

}