Computer Science Concepts and Code Snippets

# 1. Conversion Between Number Bases

## a. Convert (1012) base 2 to base 10

This problem seems to involve a typo as binary can only contain the digits 0 and 1. Please check the binary number.

## b. Convert (237) base 8 to base 2

(237)\_8 = (010011111)\_2

## c. Convert (535) base 9 to base 10

(535)\_9 = (437)\_10

# 2. Logarithm Properties

• Product Rule: log\_b (xy) = log\_b x + log\_b y

• Quotient Rule: log\_b (x/y) = log\_b x - log\_b y

• Power Rule: log\_b (x^y) = y log\_b x

• Change of Base Formula: log\_b x = log\_k x / log\_k b

• Log of 1: log\_b 1 = 0

• Log of the Base: log\_b b = 1

# 3. 1's Complement and 2's Complement

1's Complement: Flip all bits (0 to 1, 1 to 0).

2's Complement: Add 1 to the 1's complement.

Used to represent negative numbers in binary.

# 4. Power of 2 Code Implementation

```cpp  
bool isPowerOfTwo(int n) {  
 return (n > 0) && ((n & (n - 1)) == 0);  
}  
```

# 5. Representing Positive and Negative Numbers

• Positive Numbers: Use standard binary.

• Negative Numbers: Use 2's complement.

# 6. Adding and Subtracting in Binary

Addition is performed like in decimal, and subtraction uses 2's complement.

# 7. Operator Precedence Table

1. () [] -> .  
2. ++ -- + - \* / %  
3. << >>  
4. < <= > >=  
5. == !=  
6. &  
7. ^  
8. |  
9. &&  
10. ||  
11. = += -= \*= /=

# 8. Leading and Trailing Zeroes

• Leading Zeroes: Zeros before the first non-zero digit.

• Trailing Zeroes: Zeros after the last non-zero digit.

# 9. Significant Zeroes

Zeros between non-zero digits or after a decimal point.

# 10. Bit Masking

Bit masking involves using a mask to manipulate specific bits within a number.

# 11. Code Implementations

## a. Count Set Bits

```cpp  
int countSetBits(int n) {  
 int count = 0;  
 while (n) {  
 count += n & 1;  
 n >>= 1;  
 }  
 return count;  
}  
```

## b. Check if ith Bit is Set

```cpp  
bool isIthBitSet(int n, int i) {  
 return (n & (1 << i)) != 0;  
}  
```

## c. Set the ith Bit

```cpp  
int setIthBit(int n, int i) {  
 return n | (1 << i);  
}  
```

## d. Unset the ith Bit

```cpp  
int unsetIthBit(int n, int i) {  
 return n & ~(1 << i);  
}  
```

## e. Generate All Possible Subsets of an Array

```cpp  
void generateSubsets(vector<int>& arr) {  
 int n = arr.size();  
 for (int i = 0; i < (1 << n); i++) {  
 vector<int> subset;  
 for (int j = 0; j < n; j++) {  
 if (i & (1 << j)) {  
 subset.push\_back(arr[j]);  
 }  
 }  
 // Process subset (e.g., print it)  
 }  
}  
```

## f. Swapping Two Numbers

```cpp  
void swap(int &a, int &b) {  
 a = a ^ b;  
 b = a ^ b;  
 a = a ^ b;  
}  
```

## g. Odd or Even

```cpp  
bool isEven(int n) {  
 return (n & 1) == 0;  
}  
```

## h. Upper to Lower Case

```cpp  
char toLowerCase(char c) {  
 return c | ' ';  
}  
```

## i. Toggle the ith Bit

```cpp  
int toggleIthBit(int n, int i) {  
 return n ^ (1 << i);  
}  
```