Exam 1 Review; bitwise operators

COSC 208, Introduction to Computer Systems, 2022-02-15

Announcements

- Exam 1: next week -- tutor hours 6:30 to 8:30 TW evenings
- Bitwise in between vs. other operators
- Project 1: progress?

Binary arithmetic

Perform the following calculations. Operands are encoded using two's complement encoding with 6 bits. For each calculation, express the result in binary and decimal, and indicate whether the result overflows, underflows, or neither.

Q1: `0b110000 + 0b111111`

Q2: `0b001111 + 0b000001`

Q3: `0b101010 + 0b100100`

Q4: `0b001000 + 0b011000`

Q5: `0b110000 + 0b010000`

Number base conversions

Perform the following conversions

Q6: `97` to 8-bit unsigned binary

Q7: `-42` to 8-bit two's complement

Q8: `0b11001100` to unsigned decimal

Q9: `0b11001100` to signed decimal

Q10: `0x27` to unsigned decimal

Q11: `0xDEAD` to 16-bit binary

Bitwise: new!!!

Apply the following bitwise operators

Q12: `0b1010 | 0b0101`

Q13: `0b1010 & 0b0101`

Q14: `~(0b1100 & 0b0110)`

Q15: `0b1000 >> 0b011`

Q16: `0b0001 << 0b0010`

Q9: `0b1111 & (~0b0010)`

Q10: `0b0000 | 0b0010`

Logical & bitwise operators

For each of the following expressions, select all operators that make the expression evaluate to true. Operands are encoded using two's complement.

Q11: `0b110000 __ 0b111111`

```
&   &&   |   ||   ^   <
```

Q12: `0b011110 __ 0b000001`

```
&   &&   |   ||   ^   <
```

Q13: `0b000000 __ 0b000000`

```
&   &&   |   ||   ^   <
```

Q14: `0b000111 __ 0b000111`

```
&   &&   |   ||   ^   <
```

Strings

QA: *The following program should ask the user to enter a word, then print the word's length and whether it is a palindrome (i.e., reads the same backward as forward). For example, if the user enters* `"kayak"` *the program should print* `"The word is 5 characters long and is a palindrome."` *However, the program contains several errors. Modify the program to correct the errors.*

```c
#include <stdio.h>

void palindrome(char word[]) {
    int i = 0;
    int j = strlen(word);
    while (i < j) {
        if (word[i] != word[j]) {
            return -1;
        }
        i++;
        j--;
    }
    return 1;
}

int main() {
    printf("Enter a word: ");
    char word[50];
    fgets(word, 50, stdin);
    word[strlen(word)-1] = '\0'; // Remove newline
    int len = strlen(word);
    printf("The word is %c characters long and is ", len);
    if (palindrome(word)) {
        printf("a palindrome.\n");
    } else {
        printf("not a palindrome.\n");
    }
}
```

QB: *Write a function called* `molecular_formula` *that takes a string containing the constituent atoms of a molecule and updates the string to contain the molecular formula. For example, the string* `"HHO"` *should be changed to* `"H2O"`*, and the string* `"HHSOOOO"` *should be changed to* `"H2SO4"`*. You can assume:*

- *Molecules will only contain elements that are represented by a single letter — e.g., a molecule may contain* `'H'` *but not* `"Na"`
- *All atoms of the same element are listed consecutively — e.g., the constituent atoms may be provided as* `"HHO"` *but not* `"HOH"`
- *The elements are listed in the order they should appear in the molecular formula — e.g., the constituent atoms* `"HHO"` *are changed to the molecular formula* `"H2O"`*, whereas the constituent atoms* `"OHH"` *are changed to the molecular formula* `"OH2"`
- *There will be at most 9 atoms of each element — e.g.,* `"H9C9"` *may occur, but* `"H10C11"` *will not occur*

Structs

QC: *Define a struct for representing a chemical element, which includes the element's:*

- *Name*
- *Chemical symbol*
- *Atomic number*
- *State (solid, liquid, or gas) at room temperature*

QD: *Write a function called* `lookup` *that takes a chemical symbol and an array containing a struct for each of the 118 elements in the periodic table. The function should return the specified element's atomic number. If the provided symbol does not correspond to a known element, the function should return -1.*