ENGN4213/6213

Digital Systems and Microprocessors

Tutorial 5 Solutions

AUSTRALIAN NATIONAL UNIVERSITY

Question 1.1.a

Step 1: Find the find the two's complement of the magnitude bits.

$$(1011010)_2 \xRightarrow{Invert\ all\ the\ bits} (0100101)_2 \xRightarrow{Then\ plus\ one} (0100110)_2 = (38)_{10}$$

Step 2: Check the sign bit.

$$sign\ bit\ is\ 1 \rightarrow it's\ a\ negative\ number$$

Therefore, the 8-bit signed integer is **-38**.

Question 1.1.b

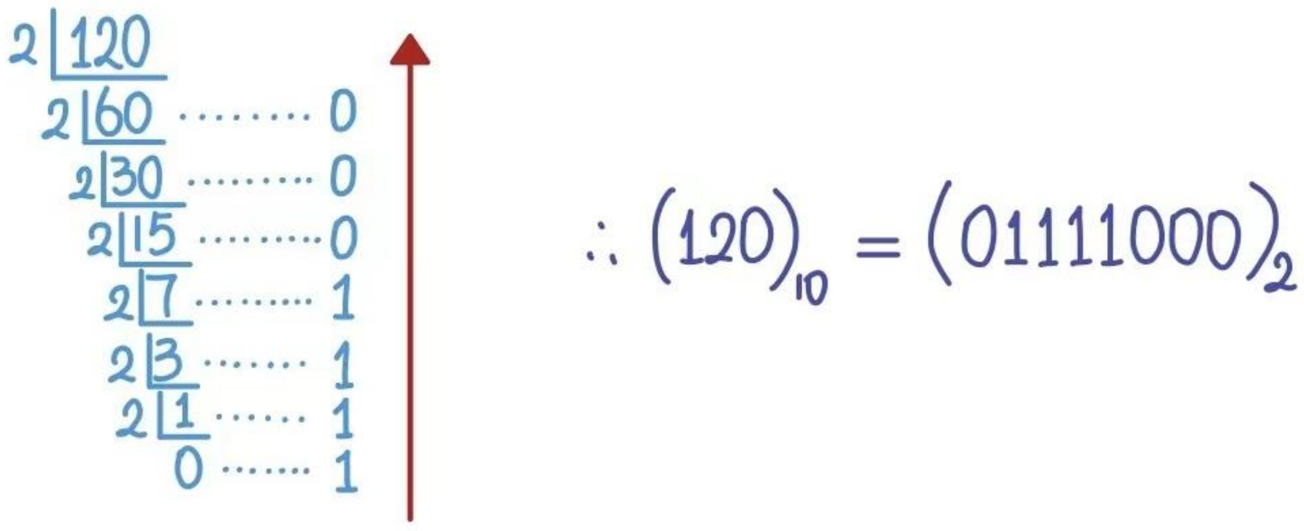
$$(11011010)_2 = 2^1 + 2^3 + 2^4 + 2^6 + 2^7 = 2 + 8 + 16 + 64 + 128 = \mathbf{(218)_{10}}$$

Question 1.1.c

$$(11011010)_2 \xRightarrow{Convert\ from\ 8-bit\ to\ 16-bit} (0000000011011010)_2 = \mathbf{(218)_{10}}$$

Question 1.2

Step 1: Convert the value into binary representation.



Step 2: Find its two's complement.

$$(01111000)_2 \xRightarrow{Invert\ all\ the\ bits} (10000111)_2 \xRightarrow{Then\ plus\ one} \mathbf{(10001000)_2 = (88)_{16}}$$

Question 1.3.a

$$c = a + b = 5 + 28 = \mathbf{33}$$

Question 1.3.b

<u>Step 1:</u> Convert the numbers into binary representation.

$$(5)_{10} = (00000101)_2$$

$$(28)_{10} = (00011100)_2$$

<u>Step 2:</u> The bitwise OR operator (|) returns a 1 in each bit position for which the corresponding bits of either or both operands are 1s.

$$(00000101)_2 \,|\, (00011100)_2 = (00011101)_2 = \mathbf{(29)_{10}}$$

Question 1.3.c

<u>Step 1:</u> Convert the numbers into binary representation.

$$(5)_{10} = (00000101)_2$$

$$(28)_{10} = (00011100)_2$$

<u>Step 2:</u> A bitwise XOR is a binary operation that takes two bit-patterns of equal length and performs the logical exclusive OR operation on each pair of corresponding bits. The result in each position is 1 if only one of the bits is 1 but will be 0 if both are 0 or are 1.

$$(00000101)_2 \wedge (00011100)_2 = (00011001)_2 = \mathbf{(25)_{10}}$$

Question 1.3.d

<u>Step 1:</u> Find the value of variable ‘d’.

Since variable ‘d’ has the type of ‘unsigned char’, its possible value can range from 0 to 255. Therefore,

$$d = e + f = 73 + 192 = 265 \xRightarrow{Overflow} 265 \,\%\, 256 = 9$$

The value of variable ‘d’ is 9.

<u>Step 2:</u> Find the value of variable ‘c’.

$$c = d + 1 = 9 + 1 = \mathbf{10}$$

<u>Step 3:</u> Double check that the value of ‘c’ is within its possible value range (-32,768 to 32,767).

Question 1.3.e

It is an infinite loop.

Again, the variable 'd' has the type of 'unsigned char', which means the maximum possible value of 'd' is 255. Therefore, variable 'd' will increase from 0 to 255 and then overflow and reset to 0. In that way, 'd' is always less than 300 and the for loop would never terminate.

Question 2

```
#include <stdio.h>

int main() {
        char name[20];
        char uid[10];

        float radius;
        float pi = 3.14159;

        printf("Enter your first name: \n");
        scanf("%s", name);

        printf("Enter your uid: \n");
        scanf("%s", uid);

        printf("Enter the radius: \n");
        scanf("%f", &radius);

        printf("Your first name %s is stored.\n", name);
        printf("Your uid %s is stored.\n", uid);

        if (radius < 0) {
                printf("Error: The radius cannot be a negative number.\n");
        } else {
                printf("The area of a circle of radius %f m is %f m^2.\n", radius, pi * radius * radius);
        }

        return 0;
}
```

Question 3

```
#include <stdio.h>
#include <float.h>

int main() {
        float numbers[5];
        float mean;
        float sum;

        float max = FLT_MIN;
        float min = FLT_MAX;

        printf("Please enter 5 integars: \n");
        for (int i = 0; i < 5; i++) {
                scanf("%f", &numbers[i]);
        }

        for (int i = 0; i < 5; i++) {
                sum += numbers[i];
                if (numbers[i] > max) {max = numbers[i];}
                if (numbers[i] < min) {min = numbers[i];}
        }
        mean = sum/5;

        printf("Mean = %f\n", mean);
        printf("Max = %f\n", max);
        printf("Min = %f\n", min);

        float sorted[5];
        for (int i = 0; i < 5; i++) {sorted[i] = numbers[i];}

        char unfinished = 1;
        while (unfinished) {
                unfinished = 0;
                for (int i = 0; i < 4; i++) {
                if (sorted[i] > sorted[i+1]) {
                        float num = sorted[i];
                        sorted[i] = sorted[i+1];
                        sorted[i+1] = num;
                        unfinished = 1;
                        }
                }
        }

        printf("You entered\tThe sorted list is\n");
        for (int i = 0; i < 5; i++) {printf("%f\t%f\n",numbers[i],sorted[i]);}

        return 0;
}
```

Question 4

```
#include <stdio.h>

int factorial(int n);

int main() {
        printf("%d\n", factorial(-1));
        return 0;
}

int factorial(int n) {
  if (n < 0) {
    printf("n should be >= 0 \n");
    return -1;
  }
        else if (n == 0) {return 1;}
        else if (n == 1) {return 1;}
        else {return n * factorial(n-1);}
}
```

Extra Question

Why do we need to declare the function before the main() function?

A function declaration tells the compiler about a function's name, return type, and parameters. On the other hand, a function definition provides the actual body of the function. Without the function declaration, when we call the function in main(), the compiler will guess the type of function output. Usually, it will assume the function output type is int by default. However, this might not always be true. For instance, if we define our function with a char output after main() without proper declaration, the compiler will still treat the function's output as int when we call the function.

Can we only define the function before the main() without declaration?

Yes, you can if you will only use the function in this particular .c file. However, if you want to use this function in some other .c files with #include, the function declaration will be necessary. Therefore, it should be good to follow the convention:

declaration -> main() -> other function definitions