실습 2

Minsoo Ryu

Operating Systems and Distributed Computing Lab.
Hanyang University

msryu@hanyang.ac.kr

목차

- 실습
  - IM.BACK
  - Bitwise Manipulation
  - IEEE 754
  - Big Number

1. IM.BACK (1)

❒ 구현 목표

- 알파벳 I, M, B, A, C, K에 **1 ~ 6** 중에서 서로 다른 값을 부여하여 수식 $\frac{I}{M} = \frac{B}{A} + \frac{C}{K}$을 만족하는 경우를 모두 찾아 $IM.BACK$ 형태로 화면에 출력하고, 또한 $\mathbf{LCM}(IM, BACK)$의 최대값을 화면에 출력하는 프로그램을 구현한다

❒ 구현 조건

- 아래의 **2**개 함수를 구현하여 프로그램에 포함시킨다

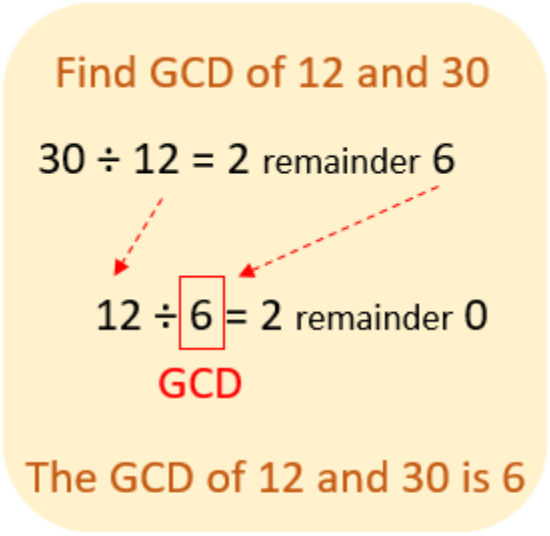
int computeGCD(int x, int y)

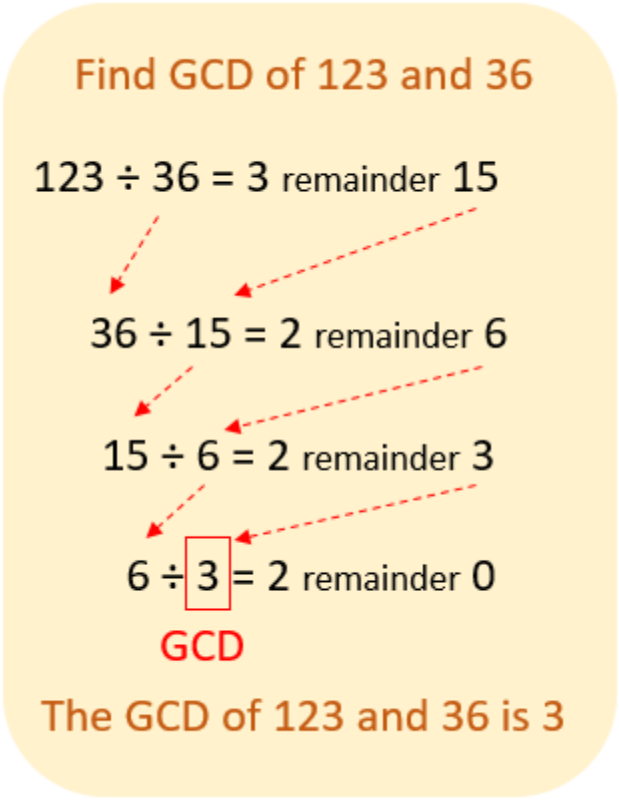
long computeLCM(int x, int y)

- **int computeGCD(int x, int y)**
  - 주어진 정수 x와 y의 최대공약수를 계산하여 리턴한다
- **long computeLCM(int x, int y)**
  - 주어진 정수 x와 y의 최소공배수를 계산하여 리턴한다

1. IM.BACK (2)

❒ 참고: GCD 계산을 위한 Euclid 호제법





1. IM.BACK (3)

❒ 실행 결과 예시

```
msryu@DESKTOP-AMQO5V8:~/Lab-Host/21_IM_BACK$ ./executable
43.1256
43.5612
56.1324
56.2413
LCM(43, 5612) = 241316
```

2. Bitwise Manipulation (1)

❒ 구현 목표

- 주어진 값에 대해 **bit** 단위로 조작하고 결과를 **binary** 형태로 출력하는 프로그램을 구현한다

❒ 구현 조건

- 아래의 **5**개 함수를 구현하여 프로그램에 포함시킨다

```
int getNthBitFromLSB(int number, int nthBit)
void setNthBitFromLSB(int* number, int nthBit)
void clearNthBitFromLSB(int* number, int nthBit)
int countOne(int number)
void printBinary(int number, int nbitsToPrint)
```

2. Bitwise Manipulation (2)

❒ 구현 조건 (계속)

- **int getNthBitFromLSB(int number, int nthBit)**
  - 주어진 정수값의 우측 끝에서부터 N번째 bit 값을 리턴한다
- **void setNthBitFromLSB(int* number, int nthBit)**
  - 주어진 정수값의 우측 끝에서부터 N번째 bit 값을 1로 바꾼다
- **void clearNthBitFromLSB(int* number, int nthBit)**
  - 주어진 정수값의 우측 끝에서부터 N번째 bit 값을 0으로 바꾼다
- **int countOne(int number)**
  - 주어진 정수값에 포함된 '1'의 개수를 리턴한다
- **void printBinary(int number, int nbitsToPrint)**
  - 주어진 정수값을 이진수(binary) 형태로 화면에 출력한다

2. Bitwise Manipulation (3)

- 실행 및 결과 확인
  - 사용자로부터 정수값을 입력받는다 **(scanf()** 함수 이용**)**
  - 주어진 정수값을 이진수**(binary)** 형태로 화면에 출력한다
    - printBinary() 실행
  - 정수값에 포함된 **‘1’**의 개수를 화면에 출력한다
    - countOne() 실행
  - 사용자로부터 특정 위치를 입력받는다 **(scanf()** 함수 이용**)**
  - 입력받은 위치의 **bit**를 **‘1’**로 바꾼다
    - setNthBitFromLSB() 실행
  - 결과를 이진수**(binary)** 형태로 화면에 출력한다
    - printBinary() 실행
  - 해당 위치의 바뀐 값을 확인하여 출력한다
    - getNthBitFromLSB() 실행
  - 입력받은 위치의 **bit**를 다시 **‘0’**으로 바꾼다
    - clearNthBitFromLSB() 실행
  - 결과를 이진수**(binary)** 형태로 화면에 출력한다
    - printBinary() 실행
  - 해당 위치의 바뀐 값을 확인하여 출력한다
    - getNthBitFromLSB() 실행

2. Bitwise Manipulation (4)

❒ 실행 결과 예시

```
msryu@DESKTOP-AMQO5V8:~/Lab/21_Bitwise_Manipulation$ ./executable
Enter an int number: 1234
1234 (in decimal)       0000 0000 0000 0000 0000 0100 1101 0010  (in binary)
The number of '1's is 5

Enter a bit position to experiment with (from LSB): 7
After setting 7th bit:  1234 (in decimal)       0000 0000 0000 0000 0000 0100 1101 0010  (in binary)
The 7th bit is 1
After clearing 7th bit: 1106 (in decimal)       0000 0000 0000 0000 0000 0100 0101 0010  (in binary)
The 7th bit is 0
```

❒ **123456789를 binary 형태로 바꾸었을 때 1은 몇 번 나타나는가?**

3. IEEE 754 (1)

❒ 구현 목표

- 주어진 **float** 값을 **IEEE 754**의 부호**(sign)**, 지수**(exponent)**, 가수**(mantissa)**로 나누어 이진수 형태로 출력하는 프로그램을 작성한다

❒ 구현 조건

- 이전 실습에서 구현한 **void printBinary(int number, int nbitsToPrint)** 함수를 재사용한다
- **void printBinary(int number, int nbitsToPrint)** 함수는 **main()** 함수가 저장된 파일이 아닌 별도의 파일에 존재해야 한다
- **Makefile**을 만들어 사용한다

3. IEEE 754 (2)

❒ 참고

- 아래의 **union** 구조를 사용하는 것을 권장

```
typedef union {
    float f;
    unsigned u;
    struct {
        unsigned mantissa : 23;
        unsigned exponent : 8;
        unsigned sign: 1;
    } raw;
} ufloat;
```

3. IEEE 754 (3)

❒ 실행 및 결과 확인

- 사용자로부터 실수값을 입력받는다 **(scanf()** 함수 이용**)**
- 주어진 실수값으로부터 부호**(sign),** 지수**(exponent),** 가수**(mantissa)**로 나누어 이진수 형태로 출력한다

❒ 실행 결과 예시

```
msryu@DESKTOP-AMQO5V8:~/Lab/23_IEEE_754$ ./executable
Enter a float number: -3948.125
Breakdown of -3948.125000:      1       1000 1010       111 0110 1100 0010 0000 0000

Sign: 1
Exponent: 1000 1010
Mantissa: 111 0110 1100 0010 0000 0000
```

❒ **3.14159**의 **exponent**와 **mantissa**는 각각 얼마인가**?**

4. Big Number (1)

❒ 구현 목표

- 화면으로부터 최대 **512** 자리의 십진수를 **2**개 입력받아 덧셈 결과를 출력하는 프로그램을 구현한다

❒ 구현 조건

- 아래의 **3**개 함수를 구현하여 프로그램에 포함시킨다

```
void convertStringToBigNumber(char string[], BigNum* bigNumber)
void printBigNumber(BigNum *bigNumber)
void addPositiveBigNumber(BigNum *first, BigNum *second, BigNum *result)
```

4. Big Number (2)

❒ 실행 결과 예시

```
msryu@DESKTOP-AMQO5V8:~/Lab-Host/29_yyy_Big_Number$ ./executable
Enter the first big integer = 1234500000
Enter the second big integer = 34567000000
1234500000 + 34567000000 = 35801500000
```

thank you!