[연습문제 - 모범답안]

[3-1] 다음 연산의 결과를 적으시오.

[연습문제]/ch3/Exercise3_1.java

```
class  Exercise3_1 {
   public static void main(String[] args) {
        int x = 2;
        int y = 5;
        char c = 'A'; // 'A'의 문자코드는 65

        System.out.println(1 + x << 33);
        System.out.println(y >= 5 || x < 0 && x > 2);
        System.out.println(y += 10 - x++);
        System.out.println(x+=2);
        System.out.println( !('A' <= c && c <='Z') );
        System.out.println('C'-c);
        System.out.println('5'-'0');
        System.out.println(c+1);
        System.out.println(++c);
        System.out.println(c++);
        System.out.println(c);
   }
}
```

[정답]

[실행결과]

```
6
true
13
5
false
2
5
66
B
B
C
```

[해설]

```
        System.out.println(1 + x << 33);
```

'1 + x << 33'는 x의 값이 2이므로 '1 + 2 << 33'가 된다. 덧셈연산자(+)보다 쉬프트연산자(<<)가 우선순위가 낮으므로 '3 << 33'이 된다. int는 32 bit이므로 33번 쉬프트 하지 않고 1번만 쉬프트 한다. '3 << 1'은 3에 '2의 1제곱' 인 2를 곱하는 것과 같은 결과를 얻으므로 '3 * 2'가 되어 결국 6을 얻는다.

```
        System.out.println(y >= 5 || x < 0 && x > 2);
```

x의 값이 2이고, y의 값이 5이므로 위의 식은 'true || false && false'가 된다. 논리연

산자 'and(&&)'는 'or(||)'보다 우선순위가 높기 때문에 'false && false'가 먼저 연산되어 'true || false'가 되고 최종결과는 true가 된다.

```
System.out.println(y += 10 - x++);
```

'y += 10 - x++'를 풀어쓰면, 'y = y + (10 - x++)'이 된다. x++은 후위형이기 때문에 x의 값이 증가되지 않은 상태에서 (10 - x)는 계산되고 x의 값은 1증가된다.
그래서 (10 - 2)로 계산이 되고 x의 값은 1증가하여 3이 된다. y의 값은 5이므로 식은 'y = 5 + (10 - 2)'가 되어 y에 13이 저장된다.

```
System.out.println(x+=2);
```

'x+=2'는 'x=x+2'와 같다. 이전의 식에서 x의 값이 1증가하였으므로 이제 x의 값은 3이다. 3에 2를 더했으므로 결과는 5가 된다.

```
System.out.println( !('A' <= c && c <='Z') );
```

!('A' <= c && c <='Z')는 문자 c가 대문자가 아닌지를 확인하는 조건식이다. 먼저 괄호안의 'A' <= c && c <='Z'가 먼저 계산되고 마지막에 이 계산결과가 논리부정연산자(!)에 의해 반대(true ↔ false)로 바뀐다. c가 'A'이므로 'A' <= 'A' && 'A' <='Z'가 되고 양쪽의 조건식이 true이므로 'true && true'의 결과인 true를 얻게 된다. 이 결과에 논리부정연산(!)을 수행하니까 true가 false로 바뀌어 최종결과는 false가 된다.

```
System.out.println('C'-c);
```

이항연산자는 피연산자가 int보다 작은 타입(byte, short, char)인 경우 int로 변환한 다음에 연산을 수행한다. c의 값이 'A'이므로 'C'-c는 'C'-'A'가 되고 'C'와 'A'는 int로 변환되어 '67 - 65'가 되고 최종결과는 2가 된다.

```
System.out.println('5'-'0');
```

'5'-'0'도 위와 같은 이유로 '53 - 48'이 되어 5를 결과로 얻는다.

```
System.out.println(c+1);
```

c+1은 c의 값이 'A'이므로 'A'+1이 되고, 이항연산자의 성질(int보다 작은 타입은 int로 변환후 연산)때문에 'A'는 문자코드 값인 65로 변환되어 '65 + 1'을 수행하여 66을 결과로 얻는다. 단지 변수 c에 저장된 값을 읽어서 수식을 계산한 것이므로 변수 c의 저장된 값에는 아무런 변화가 없다.

```
System.out.println(++c);
```

단항연산자인'++'은 이항연산자와 달리 int보다 작은 타입도 형변환을 하지 않는다.(이항연산자는 연산을 위해 '피연산자 스택(operand stack)'을 사용하는데 이 과정에서 형변환이 발생하는 것이다. 반면에 단항연산자인 증가연산자'++'은 '피연산자 스택'을 사용하지 않으므로 형변환도 발생하지 않는다.) 그래서 println은 변수 c를 숫자(int)로 출력하는 것이 아니라 문자로 출력한다. 변수 c에 저장된 문자가 'A'(실제로 저장된 것은 'A'의 문자코드인 65)이므로 문자코드의 값이 1증가되어 66('B'의 문자코드)이 변수 c에 저장된다.

변수 c에 저장된 것은 문자코드, 즉 정수값이다. println은 이 값을 타입에 따라 어떻게 출력할지를 결정한다. 만일 문자타입이면 저장된 값(문자코드)에 해당하는 문자를 출력하고 숫자라면 숫자로 출력한다.

```
System.out.println(c++);
```

단항연산자'++'이 후위형인 경우에는 println()에 의해서 변수 c가 출력된 후에 c에 저장된 값이 증가하므로 문자 'B'가 출력된 후에 변수 c의 값이 1증가해서 문자 'C'가 저장된다.

[3-2] 아래의 코드는 사과를 담는데 필요한 바구니(버켓)의 수를 구하는 코드이다. 만일 사과의 수가 123개이고 하나의 바구니에는 10개의 사과를 담을 수 있다면, 13개의 바구니가 필요할 것이다. (1)에 알맞은 코드를 넣으시오.

[연습문제]/ch3/Exercise3_2.java

```
class  Exercise3_2 {
   public static void main(String[] args) {
        int numOfApples = 123;    // 사과의 개수
        int sizeOfBucket = 10;    // 바구니의 크기(바구니에 담을 수 있는 사과의 개수)
        int numOfBucket =
            numOfApples/sizeOfBucket + (numOfApples%sizeOfBucket > 0 ? 1 : 0) ;

        System.out.println("필요한 바구니의 수 :"+numOfBucket);
   }
}
```

[실행결과]

```
13
```

[정답] **numOfApples/sizeOfBucket + (numOfApples%sizeOfBucket > 0 ? 1 : 0)**

[해설] 사과의 개수(numOfApples)를 바구니의 크기(sizeOfBucket)으로 나눗셈연산(/)을 하면 사과를 담는데 필요한 바구니의 수(numOfBucket)를 구할 수 있다. 정수간의 나눗셈연산의 특징은 반올림을 하지 않고 버림을 한다는 것이다. 예를 들어 125/10의 결과는 13이 아니라 12가 된다. 게다가 int와 int간의 이항연산결과는 int이기 때문에, 12.5와 같은 실수값 결과가 나오지 않는다.

그리고 사과의 개수(numOfApples)를 바구니의 크기(sizeOfBucket)으로 나눴을 때 나머지가 있으면 하나의 바구니가 더 필요하다. 그래서 나머지 연산자(%)를 이용해서 나눗셈연산에서 나머지가 발생하는지 확인해서, 나머지가 발생하면 바구니의 개수(numOfBucket)에 1을 더해줘야 한다.

[3-3] 아래는 변수 num의 값에 따라 '양수', '음수', '0'을 출력하는 코드이다. 삼항 연산자를 이용해서 (1)에 알맞은 코드를 넣으시오.

[Hint] 삼항 연산자를 두 번 사용하라.

[연습문제]/ch3/Exercise3_3.java

```
class  Exercise3_3 {
   public static void main(String[] args) {
        int num = 10;
        System.out.println(num > 0 ? "양수":(num < 0 ? "음수" : "0"));
   }
}
```

[실행결과]

```
양수
```

[정답] `num > 0 ? "양수":(num < 0 ? "음수" : "0")`

[설명] 삼항연산자를 사용하면 2가지 경우의 수를 처리할 수 있다. 삼항연산자에 삼항연산자를 포함시키면 3가지 경우의 수를 처리할 수 있다. num의 값이 0보다 크면, '양수'를 출력하고 끝나지만, num의 값이 0보다 작거나 같으면 괄호안의 삼항연산자가 수행된다. 여기서 num의 값이 0보다 작으면 '음수'가 출력되고, 그렇지 않으면(num의 값이 0이면) '0'이 출력된다.

[3-4] 아래는 변수 num의 값 중에서 백의 자리 이하를 버리는 코드이다. 만일 변수 num의 값이 '456'이라면 '400'이 되고, '111'이라면 '100'이 된다. (1)에 알맞은 코드를 넣으시오.

[연습문제]/ch3/Exercise3_4.java

```
class  Exercise3_4 {
   public static void main(String[] args) {
        int num = 456;
        System.out.println(num/100*100);
   }
}
```

[실행결과]

```
400
```

[정답] num/100*100

[해설] 앞의 문제에서도 설명했듯이, 나눗셈연산자는 반올림을 하지 않고 버림을 한다. 이 성질을 이용한 문제이다.

[3-5] 아래는 변수 num의 값 중에서 일의 자리를 1로 바꾸는 코드이다. 만일 변수 num의 값이 333이라면 331이 되고, 777이라면 771이 된다. (1)에 알맞은 코드를 넣으시오.

[연습문제]/ch3/Exercise3_5.java

```
class  Exercise3_5 {
   public static void main(String[] args) {
        int num = 333;
        System.out.println(num/10*10+1);
   }
}
```

[실행결과]

```
331
```

[정답] num/10*10+1

[해설] 앞의 문제를 약간 더 응용한 문제이다. 이미 다 설명한 내용이므로 자세한 설명은 생략한다.

[3-6] 아래는 변수 num의 값보다 크면서도 가장 가까운 10의 배수에서 변수 num의 값을 뺀 나머지를 구하는 코드이다. 예를 들어, 24의 크면서도 가장 가까운 10의 배수는 30이다. 19의 경우 20이고, 81의 경우 90이 된다. 30에서 24를 뺀 나머지는 6이기 때문에 변수 num의 값이 24라면 6을 결과로 얻어야 한다. (1)에 알맞은 코드를 넣으시오.

[Hint] 나머지 연산자를 사용하라.

[연습문제]/ch3/Exercise3_6.java

```
class  Exercise3_6 {
   public static void main(String[] args) {
        int num = 24;
        System.out.println(10 - num%10);
   }
}
```

[실행결과]

```
6
```

[정답] `10 - num%10`

[해설] 고민을 좀 해봐야하는 문제인데 답을 보고나면 너무 간단해서 허탈할 지도 모르겠다. 이 문제를 풀기위해 여러 연산자를 놓고 고민을 해보는 것에 의미가 있다고 생각해서 넣게 되었다. 다른 정답으로는 '(num/10+1)*10 - num'이 있다.

[3-7] 아래는 화씨(Fahrenheit)를 섭씨(Celcius)로 변환하는 코드이다. 변환공식이 'C = 5/9 ×(F - 32)'라고 할 때, (1)에 알맞은 코드를 넣으시오. 단, 변환 결과값은 소수점 셋째자리에서 반올림해야한다.(Math.round()를 사용하지 않고 처리할 것)

[연습문제]/ch3/Exercise3_7.java

```
class  Exercise3_7 {
   public static void main(String[] args) {
        int fahrenheit = 100;
        float celcius = (int)((5/9f * (fahrenheit - 32))*100 + 0.5) / 100f;

        System.out.println("Fahrenheit:"+fahrenheit);
        System.out.println("Celcius:"+celcius);
   }
}
```

[실행결과]

```
Fahrenheit:100
Celcius:37.78
```

[정답] `(int)((5/9f * (fahrenheit - 32))*100 + 0.5) / 100f`

[해설] 먼저 화씨를 섭씨로 바꾸는 공식은 '5/9f * (fahrenheit - 32)'이다. 5/9의 결과는 0이기 때문에 두 피연산자 중 어느 한 쪽을 반드시 float나 double로 해야만 실수형태의 결과를 얻을 수 있다. 그래서 정수형 리터럴인 9대신 float타입의 리터럴인 9f를 사용하였다. 소수점 셋째자리에서 반올림을 하려면 다음의 과정을 거쳐야한다.

1. 값에 100을 곱한다.
   37.77778 * 100
2. 1의 결과에 0.5를 더한다.
   3777.778 + 0.5 → 3778.278
3. 2의 결과를 int타입으로 변환한다.
   (int)3778,278 → 3778
4. 3의 결과를 100f로 나눈다.(100으로 나누면 소수점 아래의 값을 잃는다.)
   3778 / 100f → 37.78

[3-8] 아래 코드의 문제점을 수정해서 실행결과와 같은 결과를 얻도록 하시오.

[연습문제]/ch3/Exercise3_8.java

```
class  Exercise3_8 {
  public static void main(String[] args) {
      byte a = 10;
      byte b = 20;
      byte c = (byte)(a + b);

      char ch = 'A';
      ch = (char)(ch + 2);

      float f = 3 / 2f;
      long l = 3000 * 3000 * 3000L;

      float f2 = 0.1f;
      double d = 0.1;

      boolean result = (float)d==f2;

      System.out.println("c="+c);
      System.out.println("ch="+ch);
      System.out.println("f="+f);
      System.out.println("l="+l);
      System.out.println("result="+result);
  }
}
```

[실행결과]

```
c=30
ch=C
f=1.5
l=27000000000
result=true
```

[정답]

```
byte c = a + b;  → byte c = (byte)(a + b);
ch = ch + 2;     → ch = (char)(ch + 2);
float f = 3 / 2; → float f = 3 / 2f;
long l = 3000 * 3000 * 3000; → long l = 3000 * 3000 * 3000L;
boolean result = d==f2; → boolean result = (float)d==f2;
```

[해설] 이항연산은 두 피연산자의 타입을 일치시킨 후 연산을 수행한다는 것,그리고 int보다 작은 타입은 int로 자동변환한다는 것은 반드시 기억하고 있어야하는 중요한 내용이다.

```
byte c = a + b;  → byte c = (byte)(a + b);
```

변수 a와 b는 모두 byte타입이므로 이항연산인 덧셈연산을 하기 전에 int타입으로 자동형변환된다. int와 int의 덧셈이므로 연산결과는 int가 된다. int타입의 값(4 byte)을 byte타입의 변수(1 byte)에 담아야하므로 형변환을 해주어야한다.

```
ch = ch + 2;     → ch = (char)(ch + 2);
```

이것도 이전의 경우와 마찬가지이다. char타입이 덧셈연산의 과정을 거치면서 int타입으로 변환되므로 char타입의 변수에 저장하기 위해서는 char타입으로 형변환을 해주어야한다.

```
float f = 3 / 2; → float f = 3 / 2f;
```

int와 int의 연산결과는 int이기 때문에 3/2의 결과는 1이 된다. 연산결과를 실수로 얻고 싶으면, 적어도 두 피연산자 중 한 쪽이 실수타입(float와 double중의 하나)이어야한다.

```
long l = 3000 * 3000 * 3000; → long l = 3000 * 3000 * 3000L;
```

int*int*int의 결과는 int이므로 int타입의 최대값인 약 2*10의 9제곱을 넘는 결과는 오버플로우가 발생하여 예상한 것과는 다른 값을 얻는다. 그래서 피연산자 중 적어도 한 값은 long타입이어야 최종결과를 long타입의 값으로 얻기 때문에 long타입의 접미사 'L'을 붙여서 long타입의 리터럴로 만들어줘야 한다.

```
boolean result = d==f2; → boolean result = (float)d==f2;
```

비교연산자도 이항연산자이므로 연산 시에 두 피연산자의 타입을 맞추기 위해 형변환이 발생한다. 그래서 double과 float의 연산은 double과 double의 연산으로 자동형변환 되는데, 실수는 정수와 달리 근사값으로 표현을 하기 때문에 float를 double로 형변환했을 때 오차가 발생할 수 있다.

그래서 float값을 double로 형변환하기 보다는 double값을 유효자리수가 적은 float로 형변환해서 비교하는 것이 정확한 결과를 얻는다.

[3-9] 다음은 문자형 변수 ch가 영문자(대문자 또는 소문자)이거나 숫자일 때만 변수 b의 값이 true가 되도록 하는 코드이다. (1)에 알맞은 코드를 넣으시오.

[연습문제]/ch3/Exercise3_9.java

```
class  Exercise3_9 {
   public static void main(String[] args) {
        char ch = 'z';
        boolean b = ('a' <= ch && ch <= 'z') || ('A' <= ch && ch <= 'Z') || ('0' <= ch && ch <= '9');

        System.out.println(b);
   }
}
```

[실행결과]

```
true
```

[정답] ('a' <= ch && ch <= 'z') || ('A' <= ch && ch <= 'Z') || ('0' <= ch && ch <= '9') 괄호는 생략해도 된다.

[3-10] 다음은 대문자를 소문자로 변경하는 코드인데, 문자 ch에 저장된 문자가 대문자인 경우에만 소문자로 변경한다. 문자코드는 소문자가 대문자보다 32만큼 더 크다. 예를 들어 'A'의 코드는 65이고 'a'의 코드는 97이다. (1)~(2)에 알맞은 코드를 넣으시오.

[연습문제]/ch3/Exercise3_10.java

```
class  Exercise3_10 {
   public static void main(String[] args) {
        char ch = 'A';

        char lowerCase = ('A' <= ch && ch <= 'Z') ? (char)(ch+32): ch;

        System.out.println("ch:"+ch);
        System.out.println("ch to lowerCase:"+lowerCase);
   }
}
```

[실행결과]

```
ch:A
ch to lowerCase:a
```

[정답] `('A' <= ch && ch <= 'Z'), (char)(ch+32)`

[해설] 대문자인 경우에만 문자코드의 값을 32만큼 증가시키면 소문자가 된다. 문자 ch가 대문자인지를 확인하는 조건식은 **'A' <= ch && ch <= 'Z'**이고, 문자 ch의 문자코드를 32증가시키기 위해서는 덧셈연산을 해야하는데, 이 때 덧셈연산의 결과가 int이므로 char타입으로의 형변환이 필요하다.