



1. 다음 C언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.)



```
#include <stdio.h>
main() {
    int s, el = 0;
    for (int i = 6; i <= 30; i++) {
        s = 0;
        for (int j = 1; j <= i / 2; j++)
            if (i % j == 0)
                s = s + j;
        if (s == i)
            el++;
    }
    printf("%d", el);
}
```

답 : 2

[해설]

어떤 정수의 약수 중 자신을 제외한 약수를 모두 합하면 자신과 같아지는 수가 있다. 예를 들어 6의 약수 1, 2, 3, 6 중 6을 제외한 1, 2, 3을 더하면 6이 되어 자신과 같아진다. 다음은 6부터 30까지의 정수 중 이러한 약수를 갖는 수를 찾아 출력하는 알고리즘이다.

```
#include <stdio.h>
main( ) {
    ❶ int s, el = 0;
    ❷ for (int i = 6; i <= 30; i++) {
    ❸     s = 0;
    ❹     for (int j = 1; j <= i / 2; j++)
    ❺         if (i % j == 0)
    ❻             s = s + j;
    ❼     if (s == i)
    ❽         el++;
    }
    ❾ printf("%d", el);
}
```

- ❶ 정수형 변수 s, el을 선언하고, el을 0으로 초기화한다.
- ❷ 반복 변수 i가 6부터 1씩 증가하면서 30보다 작거나 같은 동안 ❸~❽번을 반복 수행한다.
- ❸ s에 0을 저장한다.

- ④ 반복 변수 j 가 1부터 1씩 증가하면서 $i/2$ 보다 작거나 같은 동안 ⑤, ⑥번을 반복 수행한다.
- ⑤ i 를 j 로 나눈 나머지가 0이면 ⑥번으로 이동하고, 아니면 현재 반복문의 처음인 ④번으로 이동한다.
- ⑥ s 에 j 의 값을 누적시킨다. 구해진 약수를 더하는 과정이다.
- ⑦ s 와 i 의 값이 같으면 약수를 모두 더한 값과 자신이 같은 수를 찾은 것이므로, ⑧번으로 이동하고, 아니면 현재 반복문의 처음인 ②번으로 이동한다.
- ⑧ ' $el = el + 1$;'과 동일하다. 약수를 모두 더한 값과 자신이 같은 수의 개수를 누적시키는 과정이다. 반복문 실행에 따른 변수들의 변화는 다음과 같다.

i	j	s	el
6		0	0
	1	1	
	2	3	
	3	6	
	4		1
7		0	
	1	1	
	2		
	3		
	4		
⋮	⋮	⋮	⋮
28		0	
	1	1	
	2	3	
	3		
	4	7	
	5		
	6		
	7	14	
	⋮	⋮	⋮
	14	28	
	15		2
⋮	⋮	⋮	⋮
31			

- ⑨ el 의 값 2를 정수로 출력한다.

결과 2

2. 다음 JAVA로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.)



```
public class Test {
    public static void main(String[] args) {
        int r = 0;
        for (int i = 1; i < 999; i++) {
            if (i % 3 == 0 && i % 2 == 0)
                r = i;
        }
        System.out.print(r);
    }
}
```

답 : 996

[해설]

문제의 코드는 1부터 998까지의 숫자 중 3과 2로 나누었을 때 나머지가 0인, 즉 6의 배수이면서 가장 큰 수를 구하는 알고리즘입니다.

```
public class Test {
    public static void main(String[] args) {
        ❶ int r = 0;
        ❷ for (int i = 1; i < 999; i++) {
            ❸ if (i % 3 == 0 && i % 2 == 0)
            ❹ r = i;
        }
        ❺ System.out.print(r);
    }
}
```

- ❶ 정수형 변수 r을 선언하고 0으로 초기화한다.
 - ❷ 반복 변수 i가 1부터 1씩 증가하면서 999보다 작은 동안 ❸, ❹번을 반복 수행한다.
 - ❸ i를 3과 2로 나눈 나머지가 모두 0이면 ❹번으로 이동하고, 아니면 반복문의 시작인 ❷번으로 이동한다.
 - ❹ r에 i의 값을 저장한다.
- 반복문 실행에 따른 변수들의 변화는 다음과 같다.

i	i%3	i%2	r
			0
1	1	1	
2	2	0	
3	0	1	
4	1	0	
5	2	1	
6	0	0	6
7	1	1	

⋮	⋮	⋮	⋮
995	2	1	
996	0	0	996
997	1	1	
998	2	0	
999	0	1	

⑤ r의 값을 출력한다.

결과 996

시나공

3. 다음 Java로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.)



```
public class Test {  
    public static void main(String[] args) {  
        int i = 3, k = 1;  
        switch(i) {  
            case 1: k++;  
            case 2: k -= 3;  
            case 3: k = 0;  
            case 4: k += 3;  
            case 5: k -= 10;  
            default: k--;  
        }  
        System.out.print(k);  
    }  
}
```

답 : -8

[해설]

```
public class Test {  
    public static void main(String[] args) {  
        ① int i = 3, k = 1;  
        ② switch(i) {  
            case 1: k++;  
            case 2: k -= 3;  
            ③ case 3: k = 0;  
            ④ case 4: k += 3;  
            ⑤ case 5: k -= 10;  
            ⑥ default: k--;  
        }  
        ⑦ System.out.print(k);  
    }  
}
```

- ① 정수형 변수 i, k를 선언하고, 각각 3과 1로 초기화한다.
- ② i의 값 3에 해당하는 숫자를 찾아간다. 'case 3' 문장으로 이동한다.
- ③ k에 0을 저장한다. → **k = 0**
※ switch 문을 종료하는 break가 없으므로 ④, ⑤, ⑥번을 모두 수행하고 ⑦번으로 이동한다.
- ④ 'k = k + 3;'과 동일하다. k의 값에 3을 더한다. → **k = 3**
- ⑤ 'k = k - 10;'과 동일하다. k의 값에서 10을 뺀다. → **k = -7**
- ⑥ 'k = k - 1;'과 동일하다. k의 값에서 1을 뺀다. → **k = -8**
- ⑦ k의 값을 출력한다.

결과 **-8**

4. 다음은 정수를 역순으로 출력하는 C언어 프로그램이다. 예를 들어 1234의 역순은 4321이다. 단, 1230 처럼 0으로 끝나는 정수는 고려하지 않는다. 프로그램을 분석하여 괄호(①~③)에 들어갈 알맞은 연산자를 쓰시오.



```
#include <stdio.h>
int main() {
    int number = 1234;
    int div = 10, result = 0;

    while (number ( ① ) 0) {
        result = result * div;
        result = result + number ( ② ) div;
        number = number ( ③ ) div;
    }
    printf("%d", result);
}
```

답

- ① **!= 또는 >**
- ② **%**
- ③ **/**

[해설]

```
#include <stdio.h>
int main() {
    ① int number = 1234;
    ② int div = 10, result = 0;

    ③ while (number != 0) {
        ④ result = result * div;
        ⑤ result = result + number % div;
        ⑥ number = number / div;
    }
    ⑦ printf("%d", result);
}
```

- ① 정수형 변수 number를 선언하고 1234로 초기화한다.
 - ② 정수형 변수 div와 result를 선언하고 각각 10과 0으로 초기화한다.
 - ③ number가 몫의 역할을 하므로 0이 될 때까지 ④~⑥번을 반복 수행한다.
 - ④ 새로운 나머지를 더하기 전에 기존의 나머지가 저장된 result에 10을 곱한다.
 - ⑤ 새로운 나머지를 result에 더한다.
 - ⑥ 다음 나머지를 구하기 위해 number를 10으로 나눈다.
- 반복문 실행에 따른 변수들의 변화는 다음과 같다.

number	div	result
1234	10	0 0 4
123		40 43
12		430 432
1		4320 4321
0		

⑦ result의 값 4321을 정수로 출력한다.

결과 **4321**

시나공

5. 다음 Java로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.)



```
public class Test {
    public static void main(String[] args) {
        int w = 3, x = 4, y = 3, z = 5;
        if((w == 2 | w == y) & !(y > z) & (1 == x ^ y != z)) {
            w = x + y;
            if(7 == x ^ y != w)
                System.out.println(w);
            else
                System.out.println(x);
        }
        else {
            w = y + z;
            if(7 == y ^ z != w)
                System.out.println(w);
            else
                System.out.println(z);
        }
    }
}
```

답 : 7

[해설]

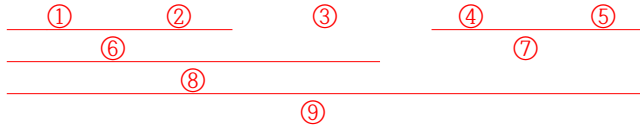
```
public class Test {
    public static void main(String[] args) {
        ❶ int w = 3, x = 4, y = 3, z = 5;
        ❷ if((w == 2 | w == y) & !(y > z) & (1 == x ^ y != z)) {
            ❸ w = x + y;
            ❹ if(7 == x ^ y != w)
            ❺ System.out.println(w);
            else
                System.out.println(x);
        } ❻
        else {
            w = y + z;
            if(7 == y ^ z != w)
                System.out.println(w);
            else
                System.out.println(z);
        } ❼
    }
}
```

❶ 정수형 변수 w, x, y, z를 선언하고 각각 3, 4, 3, 5로 초기화한다.

❷ 조건이 참이면 ❸번부터 ❹번 이전까지의 문장을, 거짓이면 ❹번 아래 else의 다음 문장부터 ❼번

이전까지의 문장을 수행한다. 연산자 우선순위에 따라 다음의 순서로 조건의 참/거짓을 확인한다.

• $(w == 2 \mid w == y) \& !(y > z) \& (1 == x \wedge y != z)$



- ① : w의 값 3과 2는 같지 않으므로 거짓(0)이다.
- ② : w의 값 3과 y의 값 3은 같으므로 참(1)이다.
- ③ : y의 값 3은 z의 값 5보다 크지 않으므로 거짓(0)이지만, 앞에 !(논리 not)가 있으므로 참(1)이다.
- ④ : 1과 x의 값 4는 같지 않으므로 거짓(0)이다.
- ⑤ : y의 값 3과 z의 값 5는 같지 않으므로 참(1)이다.

• ⑥ ① | ② : ①의 결과 0과 ②의 결과 1을 |(비트 or) 연산하면 $\begin{array}{r} 0000(0) \\ | 0001(1) \\ \hline 0001(1) \end{array}$ 이므로 결과는 1이다.

• ⑦ ④ ^ ⑤ : ④의 결과 0과 ⑤의 결과 1을 ^(비트 xor) 연산하면 $\begin{array}{r} 0000(0) \\ ^ 0001(1) \\ \hline 0001(1) \end{array}$ 이므로 결과는 1이다.

• ⑧ ⑥ & ③ : ⑥의 결과 1과 ③의 결과 1을 &(비트 and) 연산하면 $\begin{array}{r} 0001(1) \\ \& 0001(1) \\ \hline 0001(1) \end{array}$ 이므로 결과는 1이다.

• ⑨ ⑧ & ⑦ : ⑧의 결과 1과 ⑦의 결과 1을 &(비트 and) 연산하면 결과는 1이다.

∴ 최종 결과는 1이며, 1은 조건에서 참을 의미하므로 ③번으로 이동한다.

③ w에 x와 y의 합을 저장한다. (w=7)

④ 조건이 참이면 ⑤번 문장을, 거짓이면 ⑤번 아래 else 다음 문장을 수행한다. 연산자 우선순위에 따라 다음의 순서로 조건의 참/거짓을 확인한다.

• $7 == x \wedge y != w$
 ① ②
 ③

- ① : 7과 x의 값 4는 같지 않으므로 결과는 거짓(0)이다.
- ② : y의 값 3과 w의 값 7은 같지 않으므로 결과는 참(1)이다.
- ③ ① ^ ② : ①의 결과 0과 ②의 결과 1을 ^(비트 xor) 연산하면 결과는 1이다.

∴ 최종 결과는 1이며, 1은 조건에서 참을 의미하므로 ⑤번 문장을 수행한다.

⑤ w의 값 7을 출력하고 커서를 다음 줄의 처음으로 옮긴다. 모든 if문이 종료되었으므로 ⑦번으로 이동하여 프로그램을 종료한다.

결과 7

6. 다음 Python으로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.)



```
a = 100
result = 0
for i in range(1,3):
    result = a >> i
    result = result + 1
print(result)
```

답 : 26

[해설]

for문을 반복 수행할 때마다 result의 값이 누적되는 것이 아니라 새로운 값으로 치환되며, 그 값에 1을 더한다는 것을 염두에 두고 해설을 참고하세요.

```
❶ a = 100
❷ result = 0
❸ for i in range(1,3):
❹     result = a >> i
❺     result = result + 1
❻ print(result)
```

- ❶ 변수 a에 100을 저장한다.
- ❷ 변수 result에 0을 저장한다.
- ❸ 반복 변수 i가 1에서 시작하여 1씩 증가하면서 3보다 작은 동안 ❹, ❺번을 반복 수행한다.
- ❹ >>는 오른쪽 시프트 연산자이므로, a에 저장된 값을 오른쪽으로 i비트 이동시킨 다음 그 값을 result에 저장한다. 정수는 4Byte이므로 100을 4Byte 2진수로 변환하여 계산하면 된다.
- ❺ result의 값에 1을 누적시킨다.
- ※ ❸~❺ 반복 과정은 다음과 같다.

반복문 1회 수행(i = 1)

- result = a >> i

- a의 값 100을 4Byte 2진수로 표현하면 다음과 같다.

	32	31	30	29	...	9	8	7	6	5	4	3	2	1
100	0	0	0	0	...	0	0	1	1	0	0	1	0	0
					...		2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
부호 비트					...		128	64	32	16	8	4	2	1

- i는 1이므로 부호를 제외한 전체 비트를 오른쪽으로 1비트 이동시킨다. 양수이므로 패딩 비트(빈 자리)에는 0이 채워진다.

	32	31	30	29	...	9	8	7	6	5	4	3	2	1
50	0	0	0	0	...	0	0	0	1	1	0	0	1	0
					...		2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
부호 비트					...		128	64	32	16	8	4	2	1
패딩 비트					...									

- 이동된 값을 10진수로 변환하면 50이다. result에는 50이 저장된다.

- result = result + 1

- result의 값 50에 1을 더하면 result는 51이 된다.

반복문 2회 수행(i = 2)

- **result = a >> i**

- i는 2이므로 a의 값 100을 오른쪽으로 2비트 이동시킨다. 양수이므로 패딩 비트(빈자리)에는 0이 채워진다.

	32	31	30	29	...	9	8	7	6	5	4	3	2	1
25	0	0	0	0	...	0	0	0	0	1	1	0	0	1
	부호 비트		패딩 비트		...		2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
					...		128	64	32	16	8	4	2	1

- 이동된 값을 10진수로 변환하면 25다. result에는 25가 저장된다.

- **result = result + 1**

- result의 값 25에 1을 더하면 result는 26이 된다.

⑥ result의 값 26을 출력한다.

결과 **26**

시나공

7. 다음 Java로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.)



```
public class Test {
    public static void main(String[] args) {
        int j, i;
        for (j = 0, i = 0; i <= 5; i++) {
            j += i;
            System.out.print(i);
            if (i == 5) {
                System.out.print("=");
                System.out.print(j);
            }
            else
                System.out.print("+");
        }
    }
}
```

답 : 0+1+2+3+4+5=15

[해설]

```
public class Test {
    public static void main(String[] args) {
        ①      int j, i;
        ②      for (j = 0, i = 0; i <= 5; i++) {
        ③          j += i;
        ④          System.out.print(i);
        ⑤          if (i == 5) {
        ⑥              System.out.print("=");
        ⑦              System.out.print(j);
                }
            else
        ⑧          System.out.print("+");
        }
    }
}
```

- ① 정수형 변수 j, i를 선언한다.
- ② 반복 변수 i가 0부터 1씩 증가하면서 5보다 작거나 같은 동안 ③~⑧번을 반복 수행한다. i가 0으로 초기화될 때 j도 0으로 초기화된다.
- ③ 'j = j + i;'와 동일하다. j에 i의 값을 누적시킨다.
- ④ i의 값을 출력한다.
- ⑤ i가 5와 같으면 ⑥, ⑦번을 수행하고, 아니면 ⑧번을 수행한다.
- ⑥ =을 출력한다.

⑦ j의 값을 출력한다.

⑧ +를 출력한다.

※ 반복문 실행에 따른 변수들의 변화는 다음과 같다.

i	j	출력
0	0	0+
1	1	0+1+
2	3	0+1+2+
3	6	0+1+2+3+
4	10	0+1+2+3+4+
5	15	0+1+2+3+4+5=15
6		

시나공

8. 다음 Python으로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.)



```
lol = [[1, 2, 3], [4, 5], [6, 7, 8, 9]]
print(lol[0])
print(lol[2][1])
for sub in lol:
    for item in sub:
        print(item, end=' ')
    print()
```

답 :

[1, 2, 3]

7

1 2 3

4 5

6 7 8 9

[해설]

```
❶ lol = [[1, 2, 3], [4, 5], [6, 7, 8, 9]]
❷ print(lol[0])
❸ print(lol[2][1])
❹ for sub in lol:
❺     for item in sub:
❻         print(item, end=' ')
❼         print()
```

❶ 다음과 같은 행과 열을 갖는 2차원 리스트 lol이 선언되고

	[0][0]	[0][1]	[0][2]	
리스트 lol	[1][0]	[1][1]		
	[2][0]	[2][1]	[2][2]	[2][3]

다음과 같이 초기화된다.

리스트 lol	1	2	3
	4	5	
	6	7	8

❷ 리스트 lol의 0번째 행을 출력한다. 2차원 리스트에서 각각의 행은 1차원 리스트이므로 0번째 행의 요소들을 리스트 형태로 출력한다. 이어서 커서를 다음 줄의 처음으로 옮긴다.

결과 [1, 2, 3]

❸ lol[2][1]의 값을 출력한 후 커서를 다음 줄의 처음으로 옮긴다.

결과 [1, 2, 3]
7

❹ 리스트 lol의 행 수만큼 ❺~❼번을 반복 수행한다.

- sub : 리스트 lol의 각 행이 일시적으로 저장될 변수를 선언한다. sub는 1차원 리스트로 선언된다.
- lol : 리스트의 이름을 입력한다. lol 리스트가 3행이므로 각 행을 sub에 저장하면서 ❺~❼번을 3

회 수행한다.

⑤ 리스트 sub의 요소 수만큼 ⑥번을 반복 수행한다.

- **item** : 리스트 sub의 각 요소가 일시적으로 저장될 변수를 선언한다.
- **sub** : 리스트의 이름을 입력한다. sub 리스트가 차례로 3개, 2개, 4개의 요소를 가지므로 각 요소를 item에 저장하면서 ⑥번을 3회, 2회, 4회 수행한다.

⑥ item의 값을 출력하고 공백을 한 칸 띄운다.

⑦ 커서를 다음 줄의 처음으로 옮긴다.

※ 반복문 실행에 따른 변수들의 값의 변화는 다음과 같다.

sub[]	item	출력
[1, 2, 3]	1	
	2	1 2 3
	3	
[4, 5]	4	1 2 3
	5	4 5
[6, 7, 8, 9]	6	1 2 3
	7	4 5
	8	6 7 8 9
	9	

시나공

9. 다음은 변수 n에 저장된 10진수를 2진수로 변환하여 출력하는 Java 프로그램이다. 프로그램을 분석하여 괄호(①, ②)에 들어갈 알맞은 답을 쓰시오.



```
public class Test {
    public static void main(String[] args) {
        int a[] = new int[8];
        int i = 0;
        int n = 10;
        while( ( ① ) ) {
            a[i++] = ( ② );
            n /= 2;
        }
        for(i = 7; i >= 0; i--)
            System.out.print(a[i]);
    }
}
```

답

- ① $n > 0$
- ② $n \% 2$

[해설]

```
public class Test {
    public static void main(String[] args) {
        ①      int a[] = new int[8];
        ②      int i = 0;
        ③      int n = 10;
        ④      while(n > 0) {
        ⑤          a[i++] = n % 2;
        ⑥          n /= 2;
        }
        ⑦      for(i = 7; i >= 0; i--)
        ⑧          System.out.print(a[i]);
    }
}
```

① 8개의 요소를 갖는 정수형 배열 a를 선언한다.

배열 a

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

※ Java는 배열 선언 시 초기화를 하지 않아도 자동으로 0으로 초기화된다.

② 정수형 변수 i를 선언하고 0으로 초기화한다.

③ 정수형 변수 n을 선언하고 10으로 초기화한다.

④ n이 0보다 큰 동안 ⑤~⑥번을 반복 수행한다.

⑤ i++은 후치 증가 연산자이므로, a[i]에 n을 2로 나눈 나머지를 저장한 후, i의 값에 1을 더한다.

⑥ 'n = n / 2;'와 동일하다. n을 2로 나눈 값을 n에 저장한다.

반복문 실행에 따른 변수들의 값의 변화는 다음과 같다.

i	n	n % 2	a[8]
0	10		
1	5	0	0 0 0 0 0 0 0 0
2	2	1	0 1 0 0 0 0 0 0
3	1	0	0 1 0 0 0 0 0 0
4	0	1	0 1 0 1 0 0 0 0

⑦ 반복 변수 i가 7에서 시작하여 1씩 감소하면서 0보다 크거나 같은 동안 ⑧번을 반복 수행한다.

⑧ a[i]의 값을 출력한다.

반복문 실행에 따른 변수들의 값의 변화는 다음과 같다.

배열 a 0 1 0 1 0 0 0 0

i	출력
7	0
6	00
5	000
4	0000
3	00001
2	000010
1	0000101
0	00001010
-1	

시나공

10. 다음 Java로 구현된 프로그램을 분석하여 괄호(①, ②)에 들어갈 알맞은 답을 쓰시오.



```
public class Test {
    public static void main(String []args) {
        int ary[][] = new int[( ① )][( ② )];
        int n = 1;
        for(int i = 0; i < 3; i++) {
            for(int j = 0; j < 5; j++) {
                ary[i][j] = j * 3 + i + 1;
                System.out.print(ary[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

답

- ① 3
- ② 5

[해설]

```
public class Test {
    public static void main(String []args) {
        ① int ary[][] = new int[3][5];
        ② int n = 1;
        ③ for(int i = 0; i < 3; i++) {
            ④ for(int j = 0; j < 5; j++) {
                ⑤ ary[i][j] = j * 3 + i + 1;
                ⑥ System.out.print(ary[i][j] + " ");
            }
            ⑦ System.out.println();
        }
    }
}
```

- ① 3행 5열의 요소를 갖는 정수형 2차원 배열 ary를 선언한다.
- ② 정수형 변수 n을 선언하고 1로 초기화한다.
- ③ 반복 변수 i가 0에서 시작하여 1씩 증가하면서 3보다 작은 동안 ④~⑦번을 반복 수행한다.
- ④ 반복 변수 j가 0에서 시작하여 1씩 증가하면서 5보다 작은 동안 ⑤~⑥번을 반복 수행한다.
- ⑤ ary[i][j]에 $j*3+i+1$ 을 연산한 값을 저장한다.
- ⑥ ary[i][j]의 값을 출력하고 공백을 한 칸 띄운다.
- ⑦ 커서를 다음 줄의 처음으로 옮긴다.

※ 반복문 실행에 따른 변수들의 값의 변화는 다음과 같다.

n	i	j	ary[3][5]	출력															
1	0	0 1 2 3 4 5	<table> <tr><td>1</td><td>4</td><td>7</td><td>10</td><td>13</td></tr> <tr><td></td><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td><td></td></tr> </table>	1	4	7	10	13											1 4 7 10 13
1	4	7	10	13															
	1	0 1 2 3 4 5	<table> <tr><td>1</td><td>4</td><td>7</td><td>10</td><td>13</td></tr> <tr><td>2</td><td>5</td><td>8</td><td>11</td><td>14</td></tr> <tr><td></td><td></td><td></td><td></td><td></td></tr> </table>	1	4	7	10	13	2	5	8	11	14						1 4 7 10 13 2 5 8 11 14
1	4	7	10	13															
2	5	8	11	14															
	2	0 1 2 3 4 5	<table> <tr><td>1</td><td>4</td><td>7</td><td>10</td><td>13</td></tr> <tr><td>2</td><td>5</td><td>8</td><td>11</td><td>14</td></tr> <tr><td>3</td><td>6</td><td>9</td><td>12</td><td>15</td></tr> </table>	1	4	7	10	13	2	5	8	11	14	3	6	9	12	15	1 4 7 10 13 2 5 8 11 14 3 6 9 12 15
1	4	7	10	13															
2	5	8	11	14															
3	6	9	12	15															
	3																		

11. 다음 Java로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.)



```
public class Test {
    public static void main(String[] args) {
        int i = 0, c = 0;
        while (i < 10) {
            i++;
            c *= i;
        }
        System.out.println(c);
    }
}
```

답 : 0

[해설]

```
public class Test {
    public static void main(String[] args) {
        ❶ int i = 0, c = 0;
        ❷ while (i < 10) {
            ❸ i++;
            ❹ c *= i;
        }
        ❺ System.out.println(c);
    }
}
```

- ❶ 정수형 변수 i와 c를 선언하고 각각 0으로 초기화한다.
- ❷ i가 10보다 작은 동안 ❸, ❹번을 반복 수행한다.
- ❸ 'i = i + 1'과 동일하다. i의 값에 1을 누적시킨다.
- ❹ 'c = c * i'와 동일하다. c * i의 값을 c에 저장한다.
- ❺ c의 값을 화면에 출력하고 커서를 다음 줄 처음으로 옮긴다.

결과 0

※ 반복문 실행에 따른 변수들의 변화는 다음과 같다.

i	c
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0

8	0
9	0
10	0



12. 다음 Java로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.)



```
public class Test{
    public static void main(String[] args){
        int a = 0, sum = 0;
        while (a < 10) {
            a++;
            if (a%2 == 1)
                continue;
            sum +=a;
        }
        System.out.println(sum);
    }
}
```

답 : 30

[해설]

```
public class Test{
    public static void main(String[] args){
        ① int a = 0, sum = 0;
        ② while (a < 10) {
            ③ a++;
            ④ if (a%2 == 1)
            ⑤ continue;
            ⑥ sum +=a;
        }
        ⑦ System.out.println(sum);
    }
}
```

- ① 정수형 변수 a와 sum을 선언하고 각각 0으로 초기화한다.
 - ② a가 10보다 작은 동안 ③~⑥번을 반복 수행한다.
 - ③ 'a = a + 1;'과 동일하다. a의 값에 1을 누적시킨다.
 - ④ a%2, 즉 a를 2로 나눈 나머지가 1이면 ⑤번을 수행하고, 아니면 ⑥번으로 이동한다.
 - ⑤ while문의 시작점인 ②번으로 제어를 이동시킨다.
 - ⑥ 'sum = sum + a;'와 동일하다. sum에 a의 값을 누적시킨다.
- 반복문 실행에 따른 변수들의 변화는 다음과 같다.

a	sum
0	0
1	
2	2
3	

4	6
5	
6	12
7	
8	20
9	
10	30

⑦ sum의 값을 출력하고 커서를 다음 줄의 처음으로 옮긴다.

결과 30

시나공

13. 다음 JAVA로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.)



```
public class Test {
    public static void main(String[] args) {
        int result[] = new int[5];
        int arr[] = { 77, 32, 10, 99, 50 };
        for(int i = 0; i < 5; i++) {
            result[i] = 1;
            for(int j = 0; j < 5; j++)
                if(arr[i] < arr[j])
                    result[i]++;
        }
        for(int k = 0; k < 5; k++)
            System.out.print(result[k]);
    }
}
```

답 : 24513

[해설]

문제의 코드는 배열 arr에 저장된 값들의 순위를 구하여 배열 result에 저장하는 알고리즘입니다.

```
public class Test {
    public static void main(String[] args) {
        ① int result[] = new int[5];
        ② int arr[] = { 77, 32, 10, 99, 50 };
        ③ for(int i = 0; i < 5; i++) {
        ④     result[i] = 1;
        ⑤     for(int j = 0; j < 5; j++)
        ⑥         if(arr[i] < arr[j])
        ⑦             result[i]++;
        }
        ⑧ for(int k = 0; k < 5; k++)
        ⑨     System.out.print(result[k]);
    }
}
```

① 5개의 요소를 갖는 정수형 배열 result를 선언한다.

	[0]	[1]	[2]	[3]	[4]
result	0	0	0	0	0

※ Java에서는 배열을 선언하고 초기화하지 않으면 배열의 모든 요소가 0으로 초기화됩니다.

② 5개의 요소를 갖는 정수형 배열 arr을 선언하고 초기화한다.

	[0]	[1]	[2]	[3]	[4]
arr	77	32	10	99	50

③ 반복 변수 i가 0부터 1씩 증가하면서 5보다 작은 동안 ④~⑦번을 반복 수행한다.

④ 다른 점수들과 비교하기 전까지는 모든 점수의 석차는 1등이므로, result[i]에 1을 저장한다.

- ⑤ 반복 변수 j가 0부터 1씩 증가하면서 5보다 작은 동안 ⑥, ⑦번을 반복 수행한다.
- ⑥ 현재 점수(arr[i])가 비교 점수(arr[j])보다 작으면 석차를 1 증가시키기 위해 ⑦번으로 이동하고, 아니면 반복문의 시작인 ⑤번으로 이동한다.
- ⑦ 'result[i] = result[i] + 1;'과 동일하다. i번째 점수의 석차를 1씩 증가시킨다.
- 반복문 실행에 따른 변수들의 변화는 다음과 같다.

i	j	arr[i]	arr[j]	result [0] [1] [2] [3] [4]
0	0	77	77	1 0 0 0 0
	1		32	
	2		10	
	3		99	2 0 0 0 0
	4		50	
	5			
1	0	32	77	2 1 0 0 0
	1		32	2 2 0 0 0
	2		10	
	3		99	2 3 0 0 0
	4		50	2 4 0 0 0
	5			
2	0	10	77	2 4 1 0 0
	1		32	2 4 2 0 0
	2		10	2 4 3 0 0
	3		99	2 4 4 0 0
	4		50	2 4 5 0 0
	5			
3	0	99	77	2 4 5 1 0
	1		32	
	2		10	
	3		99	
	4		50	
	5			
4	0	50	77	2 4 5 1 1
	1		32	2 4 5 1 2
	2		10	
	3		99	
	4		50	2 4 5 1 3
	5			
5				

- ⑧ 반복 변수 k가 0부터 1씩 증가하면서 5보다 작은 동안 ⑨번을 반복 수행한다.
- ⑨ result[k]의 값을 출력한다.

결과 24513

14. 다음 Java로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.)



```
public class Test {
    public static void main(String[] args) {
        int aa[][] = { {45, 50, 75},
                        {89} };
        System.out.println(aa[0].length);
        System.out.println(aa[1].length);
        System.out.println(aa[0][0]);
        System.out.println(aa[0][1]);
        System.out.println(aa[1][0]);
    }
}
```

답 :

3
1
45
50
89

[해설]

```
public class Test {
    public static void main(String[] args) {
        ❶ int aa[][] = { {45, 50, 75},
                        {89} };
        ❷ System.out.println(aa[0].length);
        ❸ System.out.println(aa[1].length);
        ❹ System.out.println(aa[0][0]);
        ❺ System.out.println(aa[0][1]);
        ❻ System.out.println(aa[1][0]);
    }
}
```

❶ 4개의 요소를 갖는 정수형 2차원 배열 aa를 선언한다.

	aa[0][0]	aa[0][1]	aa[0][2]
aa	45	50	75
	89		
	aa[1][0]		

❷ aa[0] 배열의 길이 3을 출력하고 커서를 다음 줄의 처음으로 옮긴다.

- **length** : length는 배열 클래스의 속성으로, 배열 요소의 개수가 저장되어 있다. aa[0] 배열은 2차원 배열 aa의 첫 번째 행을 가리키는 것이므로, aa[0].length는 첫 번째 행의 요소 수 3을 가지고 있다.

결과 3

- ③ aa[1] 배열의 길이 1을 출력하고 커서를 다음 줄의 처음으로 옮긴다.

- aa[1] 배열은 2차원 배열 aa의 두 번째 행을 가리키는 것이므로, aa[1].length는 두 번째 행의 요소수 1을 가지고 있다.

결과 3
1

- ④ aa[0][0]의 값 45를 출력하고 커서를 다음 줄의 처음으로 옮긴다.

결과 3
1
45

- ⑤ aa[0][1]의 값 50을 출력하고 커서를 다음 줄의 처음으로 옮긴다.

결과 3
1
45
50

- ⑥ aa[1][0]의 값 89를 출력하고 커서를 다음 줄의 처음으로 옮긴다.

결과 3
1
45
50
89

시나공