프로그래밍언어론

1. <보기>의 BNF 표현과 같은 의미의 EBNF 표현으로 옳은 것은?

<보 기>

<id_list> ::= <s_list>
<s_list> ::= <letter> | <s_list> '_' <letter>

① <id_list> ::= <letter> | '_' <letter>
② <id_list> ::= {<letter>} '_' <letter>
③ <id_list> ::= {<letter> '_'} <s_list>
④ <id_list> ::= {<letter> '_' <letter>}
⑤ <id_list> ::= <letter> {'_' <letter>}

2. JAVA 언어의 메소드 재정의(method overriding)에 대한 내용으로 옳지 않은 것은?

① static, final, private 메소드는 재정의 할 수 없다.
② 메소드의 반환형(return type)은 일치하여야 한다.
③ 메소드의 이름이 같아야 한다.
④ 형식 매개변수의 개수는 일치하지 않아도 된다.
⑤ 상위클래스의 메소드를 재정의하여 하위클래스에 맞게 새롭게 구현할 수 있다.

3. <보기>의 C 프로그램 실행 결과는?

<보 기>

```
#include <stdio.h>

int X = 1;
int Y = 2;
int Z = 3;

void func(int X, int *A, int Z)
{
  Z += 4;
  X += 5;
  *A += 6;
  Y += 7;
}

int main( )
{
  int Y = 8;
  func(Y, &X, Z);
  printf("X = %d, Y = %d, Z = %d\n", X, Y, Z);
}
```

① X = 6, Y = 8, Z = 3
② X = 7, Y = 8, Z = 3
③ X = 6, Y = 13, Z = 7
④ X = 7, Y = 13, Z = 7
⑤ X = 1, Y = 13, Z = 7

4. 객체지향 프로그래밍 언어의 특징으로 옳지 않은 것은?

① SmallTalk는 완전한 객체지향 프로그래밍을 지원하는 첫 번째 프로그래밍 언어였다.
② C++는 객체지향 프로그래밍 언어이면서 전통적인 명령형 프로그래밍 언어 타입의 구조를 가질 수 있다.
③ JAVA에서 원시 스칼라 타입의 값들은 객체가 아니지만 배열은 객체이다.
④ Javascript의 객체는 속성-값(property-value) 쌍의 리스트를 통해 클래스를 지원하고, 이를 통해 다형성을 지원한다.
⑤ C#의 객체는 클래스와 struct를 통해 만들어지며, struct 객체는 스택-동적이고 상속을 지원하지 않는다.

5. 변수의 수명(life time)에 대한 설명으로 옳지 않은 것은?

① 변수의 수명은 기억장소를 할당받고 그 변수의 이름으로 할당된 기억장소가 더 이상 그 변수의 값을 의미하지 않을 때까지의 기간을 의미한다.
② Fortran에서 변수의 수명은 기억장소의 할당이 번역 시간에 정적으로 결정되며 프로그램의 실행이 끝난 후에 해제되므로 변수의 수명은 프로그램 수명과 같다.
③ JAVA나 C++에서 객체들은 new 연산에 의해 동적으로 생성되며, 이를 통해 프로그래머가 기억장소를 동적으로 할당받는 방법을 지원한다.
④ 기억장소를 할당받아 이름을 바인딩하는 것과 기억장소에 값을 할당하는 것은 항상 동시에 발생되며, 이를 통해 각 블록은 새로운 변수 속성을 결정한다.
⑤ Algol 60에서 변수의 지역적 수명은 블록이나 프로시저의 시작 시점에서 지역 변수들의 기억장소가 할당되는 방법을 지원한다.

6. <보기>의 JAVA 프로그램 실행 결과는?

<보 기>

```
public class Test {
  public static void main(String[] args) {
    outer: for(int i=0; i<2; i++) {
             for(int j=0; j<2; j++) {
               for(int k=0; k<2; k++) {
                 if(k>j) {
                   continue outer;
                 }
                 System.out.print(" "+(i+j));
               }
             }
           }
  }
}
```

① 0 1
② 0 1 2
③ 0 1 2 3
④ 0 0 0
⑤ 0 0 0 0

7. XML 구조에 대한 설명으로 옳지 않은 것은?

① DTD는 XML 문서의 구성을 정의한 것이고 element, attribute list, entity, notation 등을 구성요소로 갖는다.

② 모든 DTD 문서는 '<!'로 시작하고, 문서 타입을 정의하는 키워드인 DOCTYPE을 선언한다.

③ XSL은 XML 문서의 외형을 제어하고 스타일시트를 작성하기 위해 사용된다.

④ XSL은 XML 문서 요소를 추가하거나 변형하는 작업이 가능하고, 텍스트 기반 및 다양한 형식으로 변환할 수 있다.

⑤ DTD는 XML 파일을 브라우저가 인식할 수 있는 포맷으로 변형시켜 XML 데이터를 웹 문서 형식으로 표현할 수 있도록 한다.

8. JAVA 프로그래밍 언어에 대한 설명으로 옳지 않은 것은?

① 추상 클래스는 추상 메소드 외에 다른 멤버 변수나 메소드를 가질 수 없지만, 인터페이스는 추상 메소드와 상수만을 가진다.

② 추상 클래스는 클래스의 메소드와 변수의 일부분이 구현되어 있는 데 비해 인터페이스는 전혀 구현되어 있지 않다.

③ 추상 클래스를 이용하는 경우에는 단일 상속만 지원하는 데 비해 인터페이스는 다중 상속을 지원한다.

④ 인터페이스와 인터페이스에 정의된 메소드의 접근 한정자는 public만 사용해야 한다.

⑤ 다중 상속을 지원하기 위해 추상 클래스와 유사한 인터페이스를 제공하고 있다.

9. <보기1>의 C++ 프로그램 실행 결과가 <보기2>와 같을 때 ㉮ 안에 들어갈 코드로 옳은 것은?

<보 기 1>

```
#include <iostream.h>

void main()
{
  int i, j;
  for(i=0; i<5; i++)
  {
    ㉮
    cout << "\n";
  }
}
```

<보 기 2>

```
****
***
**
*
```

① for(j=0; j<i; j++) cout << "*";

② for(j=0; j<=i; j++) cout << "*";

③ for(j=5-i; j>1; j--) cout << "*";

④ for(j=5-i; j>=1; j--) cout << "*";

⑤ for(j=5-i; j>0; j--) cout << "*";

10. 매개변수를 이용하여 호출 프로그램에서 부프로그램에게 임의의 정보를 전달하는 방식과 그에 대한 설명으로 옳은 것은?

① 값 전달 방식(call by value) : 형식매개변수의 변화는 실매개변수에도 영향을 미친다.

② 참조 전달 방식(call by reference) : 실매개변수는 반드시 할당된 주소가 있는 변수로 해야 한다.

③ 주소 전달 방식(call by address) : 형식매개변수의 어떠한 변화에도 실매개변수에 영향을 미치지 않는다.

④ 값 결과 전달 방식(call by value-result) : 부프로그램을 종료할 때 실매개변수의 값을 형식 매개변수에 저장한다.

⑤ 이름 전달 방식(call by name) : 형식매개변수의 이름이 사용될 때마다 대응되는 실매개변수의 값으로 대치한다.

11. <보기>의 C 프로그램 실행 결과가 '4, ffbc'일 때 printf("%x\n", &n+1)의 실행 결과는?

<보 기>

```
#include <stdio.h>
void main()
{
  long n = 20;
  printf("%d, %x\n", sizeof(n), &n);
}
```

① ffbc ② ffbd

③ ffbe ④ ffbf

⑤ ffc0

12. <보기>의 JAVA 프로그램 실행 결과는?

<보 기>

```
class A {
  private int x = 1, y = 3;
  public void putX(int x) {this.x = x;}
  public void putY(int y) {this.y = y;}
  public int getResult() {return x<<y;}
}
class Test {
  public static void main(String[] args) {
    A a = new A();
    int arr[] = {1, 3, 5, 7};
    a.putX(arr[1]);
    int result = a.getResult();
    System.out.println(result);
  }
}
```

① 1 ② 8

③ 15 ④ 24

⑤ 27

13. HTML의 <A> 태그를 가장 올바르게 사용한 것은?

① <A src="http://assembly.go.kr">국회</A>
② <A href="http://assembly.go.kr"></A>
③ <A src="telnet:assembly.go.kr">국회</A>
④ <A href="ftp:assembly.go.kr">국회</A>
⑤ <A href="mailto:webmaster@assembly.go.kr">국회</A>

14. 선언문 사용의 특징으로 옳지 않은 것은?

① 동적 자료형 검사를 위해 선언문을 사용한다.
② 프로그램 실행 동안 변하지 않는 자료구조의 속성들을 한정한다.
③ 언어의 번역기는 선언문의 정보를 이용하여 최적화를 수행할 수 있기 때문에 프로그램 실행의 효율성을 높일 수 있다.
④ 효율적으로 주기억장치를 운영할 수 있다.
⑤ 자료구조의 생성 시기를 번역 시간에 검사할 수 있다.

15. <보기>의 JAVA 프로그램 실행 결과는?

<보 기>

```
abstract class B {
  int a=2;
  public int f(){return a+2;}
  public abstract void g(int i);
}
class C extends B {
  public int f(){return super.f()+2;}
  public void g(int i){System.out.println(i+2);}
}
class D extends C {
  public void g(int i){System.out.println(i+3);}
}
class Test {
  public static void main(String[] args) {
    C c = new D();
    c.g(c.f());
  }
}
```

① 4 ② 7 ③ 8 ④ 9 ⑤ 12

16. <보기>의 정규표현(regular expression)으로부터 생성될 수 없는 것은?

<보 기>

$(a|bc)^*d^+$

① bcabcd ② abcabcdd
③ bacd ④ aabcad
⑤ d

17. 전처리기(preprocessor)에 대한 설명으로 옳은 것은?

① 고급 언어로 작성된 원시 프로그램을 목적 프로그램으로 번역한 것으로 프로그램을 실행할 때마다 목적 프로그램을 다시 번역하지 않는다.
② 유사한 원시 코드를 매크로로 정의하고 필요할 때마다 확장하여 프로그래머의 생산성을 증가시킨다.
③ 기계어를 1:1로 대응시켜 기호화한 언어로 목적 프로그램의 생성 절차가 간단하다.
④ 목적 프로그램을 만들지 않고 원시 프로그램의 명령문을 번역하면서 바로 실행된다.
⑤ 문장 단위로 번역하여 실행하고, 전체 프로그램을 완성하지 않아도 결과를 알 수 있다.

18. 다음에 제시된 C++ 프로그램 중에서 for문이 무한 반복 실행되는 것은?

① for(int i=1; i>2 ; i++) { cout << i; }
② for(int i=1; i<2 ; i++) { cout << i; }
③ for(int i=1; i=2 ; i++) { cout << i; }
④ for(int i=1; i<2 ; ++i--) { cout << i; }
⑤ for(int i=1; ++i<2 ; i) { cout << i; }

19. 활성 레코드(activation record)에 대한 설명으로 옳지 않은 것은?

① 재귀 호출을 위해서는 정적 활성 레코드 할당이 필요하다.
② 매개변수를 위한 기억공간을 포함한다.
③ 지역변수를 위한 기억공간을 포함한다.
④ 반환 값을 위한 기억공간을 포함한다.
⑤ 프로시저 호출/반환에 필요한 정보를 저장하는 자료 구조이다.

20. 예외 처리 및 이벤트 처리에 대한 설명으로 옳지 않은 것은?

① Ada는 예외가 발생할 수 있는 코드에 지역적이며 동일한 참조 환경이 제공되기 때문에 처리기를 위한 매개 변수는 필요하지 않으며 허용되지 않는다.
② C++에서 예외 처리는 예외가 발생할 것으로 예상되는 지역에 대한 try문과 예외 처리를 위한 catch문으로 구성된다.
③ JAVA는 JVM(Java Virtual Machine)에 의해서 묵시적으로 발생하는 미리 정의한 예외의 집합을 포함하며, 예외 클래스는 Throwable 클래스의 후손 클래스이다.
④ JAVA에서 이벤트는 반드시 사용자로부터만 발생하는 것은 아니며, 타이머가 종료되거나 카운터 값이 지정된 범위를 벗어났을 때도 소프트웨어적으로 발생할 수 있다.
⑤ C++에서 임의의 클래스가 public으로 내포되어 있는 예외 처리 클래스를 포함하는 경우 이 클래스로부터 유도된 클래스는 별도의 예외 처리 클래스를 정의할 수 없으며, 기초 클래스에 포함되어 있는 예외 처리 클래스는 상속받을 수 없다.