**신입사원 기초 프로그래밍 교육**

2014-01-06 ㈜ 모비젠

**준비단계 – 개발환경 준비하기**

가. 자기 PC에 자바 개발환경을 설치한다.

 - Hello World 자바 프로그램을 짜고 실행한다.

나. 자기 PC에 파이선(Python) 개발환경을 설치한다.

 - Hello World 파이선 프로그램을 짜고 실행한다.

다. 자기 PC에 VirtualBox를 설치하고, 리눅스 가상 머신 및 Windows 가상 머신을 설치한다.

- 가상머신 이미지는 별도로 구해서 설치한다.

라. 리눅스 가상머신 및 Windows가상 머신 속에 각각 자바 및 Python 개발 환경을 설치한다.

 - 각각의 가상머신 속에서 Hello World 프로그램을 짜서 실행시킨다.

**자료형**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 자료형 | 자료형 | 기본 자료형을 설명할 수 있다  |
| 2 | 　 | 　 | 각 자료형의 한계를 설명할 수 있다. |
| 3 | 　 | 　 | 자료의 형변환에 대해서 설명할 수 있다. |

가. 1부터 100까지 곱하는 코드를 Java와 Python으로 작성하라.
 - 단순하게 코딩했을 때, 자바에서 발생하는 Exception이 무엇인지 알아내라.

나. ‘1’ + 2 라는 코드가 정상적으로 동작하는 자바/파이선 코드를 작성하라.

 - 문자 그대로 코딩했을 때, 나타나는 에러를 캡쳐하라.

 - 오류 없이 정상 동작 시켰을 때, 기대할 수 있는 서로 다른 값을 출력하고, 그 이유를 설명해 보라.

**배열(어레이)**

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | 　 | 어레이 | 다차원(2차원/3차원) 어레이를 생성하고 활용할 수 있다. |
| 5 | 　 | 　 | 어레이의 한계를 설명할 수 있다. |
| 6 | 　 | 　 | 어레이의 Insert, Delete, Replace, Slice를 할 수 있다. |
| 7 |  |  | 어레이의 Resizing을 설명할 수 있다. |

가. 2 x 3 어레이 a를 생성하고, 각각에 값을 1부터 6까지 채우는 자바 및 파이선 코드를 작성하라.

 - 어레이의 내용을 출력해 보라.

 - a[1][2]의 값을 출력하라.

나. 배열(어레이) 자료를 복사하는 코드를 자바 및 파이선으로 작성하라.

다. 1000개의 int를 갖는 어레이에 1부터 1000까지 저장하고, 그 중에 2번째 항목을 제외하고, 나머지 999개를 복사하는 코드를 효율적으로 작성해 보라. (자바, 파이선 각각)

 - array copy 개념 사용

라. 다음을 수행하는 코드를 자바 및 파이선으로 작성하라.

 - 6개인 항목의 어레이를 새로 선언하는 코드.

 - 해당 어레이의 4번째/5번째 항목을 지우고, 그 자리에 새로운 숫자 하나를 끼워 넣는 코드를 작성하라. 결과적으로 어레이의 길이가 5가 됨을 확인하라.

마. 항목 6개인 어레이 2개를 따로 선언한 다음, 그것들을 연결해서 총 12개가 들어 있는 어레이를 재구성해 볼 것. (자바, 파이선)

바. 생성 가능한 최대한의 어레이를 생성해 보라. (프로그램이 죽지 않는 범위 내에서 최대 어레이 생성) (자바, 파이선)

**리스트**

|  |  |  |  |
| --- | --- | --- | --- |
| 8 | 　 | 리스트 | 리스트와 어레이 사이의 변환을 할 수 있다. |
| 9 | 　 | 　 | 리스트에 Prepend, Append 등을 할 수 있다. |
| 10 | 　 | 　 | 리스트의 내용물의 타입을 강제할 수 있다. |
| 11 | 　 | 　 | Heterogeneous 리스트를 만들 수 있다. |
| 12 | 　 | 　 | 변형된 리스트(큐,스택,벡터,데크)의 특성을 설명할 수 있다. |
| 13 | 　 | 　 | 큐를 설명할 수 있다. |
| 14 | 　 | 　 | 우선순위 큐를 설명할 수 있다. |
| 15 | 　 | 　 | 서큘러 큐를 설명할 수 있다. |
| 16 | 　 | 　 | 스택의 기본 연산을 설명할 수 있다. |

가. 10개의 값을 가지고 있는 어레이의 내용을 그대로 리스트에 복사하는 코드를 작성하라.

 - 값이 int 인 경우와, String인 경우로 구분하여 코딩하라.

 - 자바와 파이선으로 코딩하라.

나. 리스트의 0번째 위치에 1,2,3,4,5,6,7,8,9를 차례로 넣어서, 결과적으로, 9,8,7,6,5,4,3,2,1이 들어 있는 리스트를 만들어라. (자바/파이선)

 - 홀수번째 있는 것들만 지우고, 결과적으로 짝수번째 있는 것들만 남게 해 보라.

다. [1, ‘가’, “abc”, -1.1] 과 같이 Heterogeneous한 리스트를 만들고, 내용물을 출력해 보라.

 - 자바/파이선 각각.

라. 리스트를 이용해서 push() / pop()이 지원되는 스택을 만들어 보라. (자바/파이선 각각)

마. 리스트에 최대한 많은 정수를 담아 보라. 몇 개까지 저장할 수 있는가? 메모리가 부족해서 죽을 때까지 몇 초 걸리는가?

**해쉬맵**

|  |  |  |  |
| --- | --- | --- | --- |
| 17 | 　 | 해쉬맵 | 해쉬맵에 자료를 Traverse할 수 있다. |
| 18 | 　 | 　 | 해쉬맵의 초기화 인자를 설명할 수 있다. |
| 19 | 　 | 　 | 해쉬맵의 데이터를 소팅할 수 있다. |
| 20 | 　 | 　 | 해쉬맵의 사용예를 설명할 수 있다. |
| 21 | 　 | 　 | 해쉬맵의 성능을 설명할 수 있다. |
| 22 | 　 | 　 | 해쉬맵의 다른 변형들을 설명할 수 있다. (딕셔너리, 트리맵, …) |

가. 키:밸류 쌍이 {‘가나다’: 3, ‘abc’: 4, ‘a’: 123}인 해쉬(또는 맵 또는 딕트)을 만들어 보라. (자바/파이선)

 - 만든 해쉬맵의 내용물을 모두 출력하는 코드를 작성하라.

나. {String: String} 을 담을 수 있는 해쉬맵을 하나 만든 후에, 메모리가 Full 날 때까지 랜덤하게 스트링을 생성해서 해당 해쉬맵에 값을 저장해 보라. 최대 몇 개까지 넣을 수 있는가? 그때까지 걸리는 시간은 몇 초인가? (자바/파이선)

 - 자바와 파이선 중에 누가 더 많은 데이터를 담을 수 있는가?

 - 메모리가 증가하는 모습(작업관리자?)을 그래프로 확인해 보라.

다. 해쉬맵의 키를 소팅하여 저장할 수 있는가? (자바/파이선)

 - 해쉬맵의 밸류를 소팅하여 저장할 수 있는가? (자바/파이선)

 - 키 로 소팅된 순서대로 출력해 보라.

 - 밸류로 소팅된 순서대로 출력해 보라.

**스트링**

|  |  |  |  |
| --- | --- | --- | --- |
| 23 | 　 | 스트링 | 스트링 초기화를 할 수 있다. |
| 24 | 　 | 　 | 스트링과 캐릭터의 연관관계를 설명할 수 있다. |
| 25 | 　 | 　 | 스트링을 복사하는 여러가지 방법을 알고 있다. |
| 26 | 　 | 　 | 스트링과 메모리 표현의 차이를 설명할 수 있다. |
| 27 | 　 | 　 | Immutable 스트링의 속성을 설명할 수 있다. |
| 28 | 　 | 　 | 스트링의 본질 연산을 설명할 수 있다. |
| 29 | 　 | 　 | 스트링 인코딩을 설명할 수 있다. |

가. 자바(Java) 스트링을 생성하는 여러 가지 다른 방법을 코딩해 보라.

 - 파이선 스트링도 다양한 방법으로 생성해 보라.

나. ‘한글 스트링’을 생성하고, 글자수를 카운트 하는 코드를 작성해 보라.

 - 글자수 값이 다르게 나오도록 코드를 변형시켜 보라.

다. 자바/파이선 각각, 특정 함수에 스트링을 전달한 후, 함수 속에서 스트링 값을 변형한 후에, 함수 밖에서 값이 바뀌었는지를 확인해 볼 것.

라. 스트링의 처음 2글자가 “0x”로 시작하면, hexa-decimal이라고 가정하고, 나머지 글자들을 16진수 디코딩해서 integer 숫자로 표현하는 코드를 작성하라.

마. USD 11,000 $ 와 같은 형태의 숫자 표현을 받아들이는 프로그램인데, 맞는 표현과 틀린 표현을 구별해 내는 프로그램을 작성해라. (Java, Python) 예: usD 11,00 $ 는 대소문자도 틀렸고, 자릿수가 틀렸다. USd 1 $ 는 대소문자가 틀렸다. USD10$는 맞다. USD 10,000$ 도 맞다. USD 99.9$ 도 맞다. USD 9,999.1$도 맞다.

**인코딩**

|  |  |  |  |
| --- | --- | --- | --- |
| 30 | 인코딩 | 인코딩개념 | 인코딩과 스트링/캐릭터의 연관관계를 설명할 수 있다. |
| 31 | 　 | 　 | 아스키/EUC-KR 등을 설명할 수 있다. |
| 32 | 　 | 　 | UTF-8 인코딩을 설명할 수 있고, 인코딩/디코딩을 설명할 수 있다. |
| 33 | 　 | 　 | Base64 인코딩을 설명할 수 있다. |
| 34 | 　 | 　 | 값을 유지하는 인코딩과 값이 변하는 인코딩을 구분할 수 있다. |
| 35 | 　 | 　 | 인코딩이 사용되는 응용예를 설명할 수 있다. |
| 36 | 　 | 　 | 각 인코딩에서 발생하는 문제들을 설명할 수 있다. |

가. EUC-KR 로 표현한 ‘한글’의 코드 값을 출력하도록 하라. (Python, Java)

 - UTF-8 일 때 동일한 ‘한글’의 코드 값을 출력하라. (Python, Java)

나. EUC-KR로 인코딩된 ‘한글’이 Base64로 다시 인코딩 되면 최종적으로 생기는 스트링값은 어떻게 되는지 알아보라. (Java, Python)

다. UTF-8로 표현된 ‘한글’을 Hex 코드로 인코딩하면 최종 값이 어떻게 되는지 알아 보라.

라. UTF-8로 표현된 ‘한글’을 Base64로 인코딩한 후, Hex를 인코딩한다. 그 역으로 Hex 디코딩해서, Base64 디코딩하면 UTF-8로 표현된 한글이 나오는 과정을 보여주는 자바/파이선 코드를 작성하라.

마. UTF-8로 표현된 한글 텍스트에 대해서 .substring(0, n)을 구현하라. (단, 한글이 깨지면 안됨.) (자바/파이선)

**파일 입출력**

|  |  |  |  |
| --- | --- | --- | --- |
| 37 | 파일 | 기본 | 파일 오픈/클로즈/Read/Write를 할 수 있다. |
| 38 | 　 | 　 | Stream(InputStream,OutputStream)과 Reader/Writer의 차이를 설명할 수 있다. |
| 39 | 　 | 　 | I/O 버퍼링을 이해하고 설명할 수 있다. |
| 40 | 　 | 　 | Line I/O의 성능 이슈를 설명할 수 있다. |
| 41 | 　 | 　 | I/O 에러 핸들링의 어려움을 설명할 수 있다. |
| 42 | 　 | 　 | 바이너리 파일 I/O를 설명할 수 있다. |
| 43 | 　 | 　 | 파일 동시 접근에 대해서 설명할 수 있다. |
| 44 | 　 | 　 | 파일 LOCK의 방법을 설명할 수 있다. |

가. 이미 존재하는 파일에서 10 바이트 만큼을 3번 읽어 들이는 코드를 작성하라. (자바/파이선)

나. 10줄의 텍스트가 들어 있는 파일을 메모장으로 만든 후에, 해당 파일의 3번째 줄의 텍스트를 읽어서 출력하는 코드를 작성하라. (자바/파이선)

다. 1줄의 길이가 2만 바이트인 텍스트 3줄이 들어 있는 텍스트 파일을 생성하라.

 - 자바/파이선으로 그런 텍스트 파일을 생성하는 코드를 작성하라.

라. 위의 파일에서 두번째 줄의 텍스트를 모두 점(‘.’)으로 바꾸는 프로그램을 작성하라. (자바/파이선)

- 파일의 길이는 변하지 않아야 한다.

마. 3줄의 한글/영문 혼합 문자열이 들어있는 텍스트 파일을 만들어라. 프로그램1은 두번째 줄의 바이트 수를 카운트하고, 프로그램2는 두번째 줄의 글자수를 카운트 하는 프로그램을 작성하라. (자바/파이선)

바. “무궁화꽃이피었습니다”가 기록되어 있는 텍스트 파일이 있다고 하자. 4번째 바이트에 ‘4’를 기록해 넣는 코드를 작성하라. 결과로 생긴 파일은 어떤 모양인가? (자바, 파이선)

사. 마.번의 2개의 프로그램 각각을, 파일을 열자마자 lock을 걸고, 카운트하고, 1초 쉬고, lock을 푸는 형태로 프로그램을 수정한다. 수정한 후에, 두 프로그램을 동시에 실행시키면, 어떤 일이 일어나는지 알아내라. 실제로 화면을 두 개 띄워서 2개를 동시에 실행시켜 보라. (자바/파이선)

**디렉토리**

|  |  |  |  |
| --- | --- | --- | --- |
| 45 | 　 | DIR | 파일 상태 및 아이노드 개념을 이해할 수 있다. |
| 46 | 　 | 　 | 파일 INODE에 대한 각종 연산을 설명할 수 있다. |
| 47 | 　 | 　 | 파일 아이노드구조에 따른 LOCK 개념을 설명할 수 있다. |
| 48 | 　 | 　 | 파일의 구조에 따른 성능 차이를 설명할 수 있다. |

가. 자기의 PC속에 파일이 몇 개나 있는지 개수를 구하는 프로그램을 작성하라. (자바/파이선)

나. 자기의 PC 속에 파일들을 다 뒤져서, \*.jpg 또는 \*.gif 그림 파일의 개수를 구하라.

다. 자기의 PC 속에서, 디렉토리 깊이가 3레벨이면서, 생성시간이 1년 채 안된 파일의 개수를 구하는 프로그램을 작성하라. C:\Windows 는 1레벨이다.

라. 자기 PC에 있는 \*.jpg / \*.gif / \*.png 등의 그림 파일 중에서, 파일 크기가 가장 큰 3개를 찾아내는 프로그램을 작성하라.

**네트워크 소켓**

|  |  |  |  |
| --- | --- | --- | --- |
| 49 | 네트웍 | 소켓 기본 | 서버, 클라이언트, 커넥션, 프로토콜 등의 용어를 설명할 수 있다. |
| 50 | 　 | 　 | 소켓 클라이언트의 전형적인 패턴을 쓸 수 있다. |
| 51 | 　 | 　 | 소켓 서버 코드의 전형적인 패턴을 쓸 수 있다. |
| 52 | 　 | 　 | Read/Write의 예외처리 방법이 가지는 단점을 설명할 수 있다. |
| 53 | 　 | 　 | 소켓 타임아웃을 정의할 수 있다. |

가. 소켓을 이용해서, 대문자로 변환하는 에코 서버를 만들어라. (자바/파이선)

 - 해당 포트에 대고 “abc” 라고 입력하면, “+OK ABC” 라고 응답이 나와야 한다.

나. 에코 서버에 붙어서, “a”라는 소문자를 전달하고, “+OK A”를 받아내는 클라이언트 프로그램을 만들어라. (자바/파이선)

다. 에코 서버에서 입력이 3초 이내에 들어오지 않으면 “-ERROR TIMEOUT” 이라고 출력하고 커넥션을 끊도록 코드를 수정하라. (자바/파이선)

**네트워크 소켓 – 비동기**

|  |  |  |  |
| --- | --- | --- | --- |
| 54 | 　 | 비동기 | 비동기 I/O가 어떤 것인지 설명할 수 있다. |
| 55 | 　 | 　 | 비동기 I/O의 전형적인 패턴을 쓸 수 있다. |
| 56 | 　 | 　 | 제어의 역전(Inversion of Control)을 설명할 수 있다. |

가. 포트 2개를 동시에 Listening/Accept하는 Echo 서버를 작성하라. (자바/파이선)

 - 단 스레드는 1개이어야 한다. (2개 이상이면 안됨)

 - 두 개의 Port중에 어느 쪽에 먼저 입력이 들어오더라도 바로 반응을 해야 한다.

**네트워크 소켓 – UDP**

|  |  |  |  |
| --- | --- | --- | --- |
| 57 | 　 | UDP | UDP 소켓과 TCP 소켓의 차이를 설명할 수 있다. |
| 58 | 　 | 　 | 전형적인 UDP 소켓의 코드를 짤 수 있다. (서버/클라이언트) |

가. UDP 소켓 서버로 에코 서버를 작성한다. (자바/파이선)

나. 에코 서버에 대응되는 UDP 소켓 클라이언트 프로그램을 작성한다. (자바/파이선)

**네트워크 소켓 – 상태감시**

|  |  |  |  |
| --- | --- | --- | --- |
| 59 | 　 | 상태 감시 | 소켓의 상태를 리눅스에서 모니터링할 수 있다. |
| 60 | 　 | 　 | 소켓의 전형적인 에러 패턴을 설명하고, 모니터링으로 설명할 수 있다. |
| 61 | 　 | 소켓 파라메터 | OS(Linux)에서 제공하는 소켓 파라메터 종류를 설명할 수 있다. |
| 62 | 　 | 　 | OS(Linux)에서 제공하는 소켓 파라메터 변경 방식을 설명할 수 있다. |

가. 리눅스에서 TCP 소켓 서버 및 UDP 소켓 서버를 실행시킨 후에, netstat –an 명령어로 해당 소켓의 상태를 볼 수 있다.

나. 에코 서버와 에코 클라이언트 프로그램을 이용해서 프로그램에서, 서버 및 클라이언트 각각에 대해서, .close() 를 호출하지 않는 오류가 있도록 프로그램을 바꾼다. 결과적으로 .close()가 호출되지 않으면 netstat –an 의 결과가 어떻게 표시되는지 알아본다. 클라이언트/서버 각각의 케이스를 따로 분리해서 본다.

다. 동시에 Open할 수 있는 파일/소켓의 최대 개수를 알아내는 프로그램을 짠다. 다시말해 더 이상 오픈 할 수 없을 만큼 최대한 소켓 또는 파일을 오픈해서 오류 날 때까지 가 본다. (자바/파이선) 어떤 오류가 나는 지를 확인해 본다.

**쓰레드**

|  |  |  |  |
| --- | --- | --- | --- |
| 63 | 스레드 | 스레드 | 멀티쓰레드와 멀티프로세스의 차이를 설명할 수 있다. |
| 64 | 　 | 　 | 스레드 생성 및 종료를 쓸 수 있다. |
| 65 | 　 | 　 | 스레드 간의 공유를 설명할 수 있다. |
| 66 | 　 | 　 | 스레드의 메모리 사용 및 한계를 설명할 수 있다. |
| 67 | 　 | 　 | 스레드의 성능 이슈를 설명할 수 있다. |

가. 1초에 1번씩 가위/바위 보 게임을 하는 프로그램을 짠다. 2개의 서로 다른 쓰레드가 각각 랜덤하게 가위/바위 보를 낸다. 1초에 1회씩 한다. 각각 승점을 카운트한다. 30초 지나면 승점이 각각 15대 15정도 수준에 수렴함을 보인다. (자바/파이선)

나. 1초에 10번씩 가위/바위 보 게임을 하는 프로그램을 짠다. 200개의 서로 다른 쓰레드가 랜덤하게 가위바위 보를 낸다. 이기는 쓰레드가 나올 때까지 한다. (자바/파이선) 단, 그때까지 각 쓰레드가 가위바위보를 한 횟수는 동일해야 한다.

다. 가위바위보 게임을 1초에 100회씩 한다고 하고, 최대 몇 개의 쓰레드까지 실행시킬 수 있는가? 자기 PC의 한계까지 끌어 올려서 오류가 나는 지점을 찾아보라. (자바/파이선)

**기본 구문**

|  |  |  |  |
| --- | --- | --- | --- |
| 68 | 기본구문 | IF/WHILE/FOR | 루프 구문을 정확히 설명할 수 있다. |
| 69 | 　 | 　 | FOR/WHILE/DO-WHILE을 서로 변환해서 코딩할 수 있다. |
| 70 | 　 | 　 | FOR-EACH를 설명할 수 있다. |
| 71 | 　 | 　 | 루프의 스코프에 대해서 설명할 수 있다. |
| 72 | 　 | 　 | 루프의 성능 저하요인을 설명할 수 있다. |
| 73 | 　 | 　 | 다중루프를 설명할 수 있다. |
| 74 | 　 | 　 | 다중 루프의 복잡도(성능상의 복잡도)를 설명할 수 있다. |
| 75 | 　 | 　 | 너무 깊은 다중 루프를 피하는 법을 설명할 수 있다. |

가. while 구문으로 for 구문을 대신하여 쓰는 예를 작성하라. (Java/Python)

나.for 구문과 for each 구문을 구분해서 예를 작성해 보라. (Java/Python)

다. for each 구문에서 each 에 오는 항목을 루프 내에서 값을 바꾼 후에, 루프 밖에서 어떻게 되는지 확인해 볼 것. (Java/Python)

라. 파이선 list comprehension 을 풀어서 java로 표현해 보세요.

 result = [i for i in mylist if i.attribute == None]

**암호화 / 인크립션**

|  |  |  |  |
| --- | --- | --- | --- |
| 76 | 암호화 | 인크립션 | 인크립션(암호화)와 인코딩의 차이를 설명할 수 있다. |
| 77 | 　 | 　 | 대칭키/비대칭키 암호화를 설명할 수 있다. |
| 78 | 　 | 　 | 암호화 인크립션 사용 코드를 해석할 수 있다. |

가. 자바에서 UTF-8로 인코딩된 한글 텍스트 “무궁화 꽃이 피었습니다.”를 암호화하고, 파이선으로 해당 텍스트를 복호화 하는 예제를 작성하라. (암호화 알고리듬은 알아서 선택할 것)

나. 그 역으로 파이선에서 암호화 하고, 자바에서 복호화 하는 예제를 작성하라.

다. 자바로 SHA256 으로 비밀번호를 암호화하는 코드를 작성하고, 파이선에서 해당 암호가 정상인지 아닌지 검증하는 코드를 작성하라. 암호화된 결과는 Hexadecimal 스트링으로 표현되어야 한다.

**정규식**

|  |  |  |  |
| --- | --- | --- | --- |
| 79 | 정규식 | 정규표현식 | 정규표현식의 목적을 설명할 수 있다. |
| 80 | 　 | 　 | 기본 정규표현식을 쓸 수 있다. |
| 81 | 　 | 　 | 정규표현식을 이용한 Substitution을 할 수 있다. |
| 82 | 　 | 　 | 정규표현식을 이용한 Extraction을 할 수 있다. |
| 83 | 　 | 　 | 정규표현식의 컴파일 및 매칭의 메커니즘을 설명할 수 있다. |
| 84 | 　 | 　 | 정규표현식의 성능이슈를 설명할 수 있다. |

가. 임의의 텍스트에서 이메일 주소를 추출해 내는 코드를 정규표현식을 이용해서 작성하라.

나. 주민등록번호가 들어 있는 텍스트에서 주민등록 번호 부분을 별표(‘\*’)로 마스킹하는 코드를 작성하라.

다. 임의의 텍스트에서 정규표현식을 사용하지 않고, 이메일 주소를 추출하는 코드를 작성하라.

**함수**

|  |  |  |  |
| --- | --- | --- | --- |
| 85 | 함수 | 언어의 기본 | 함수가 무엇인지 설명할 수 있다. |
| 86 | 　 | 　 | 함수의 리턴값과 인자를 설명할 수 있다. |
| 87 | 　 | 　 | 함수 인자가 가변적인 경우를 설명할 수 있다. |
| 88 | 　 | 　 | 함수의 오버라이딩을 설명할 수 있다. |
| 89 | 　 | 　 | 연산자가 함수와 같음을 설명할 수 있다. |
| 90 | 　 | 　 | 객체지향의 메서드와 함수를 구별하여 설명할 수 있다. |
| 91 | 　 | 　 | 함수의 인수 전달 방식을 설명할 수 있다. |
| 92 | 　 | 　 | 함수의 스택 동작을 설명할 수 있다. |
| 93 | 　 | 　 | 스택 메모리와 힙 메모리를 함수와 연관지어 설명할 수 있다. |
| 94 | 　 | 　 | Call-by-Value, Call-by-Reference를 설명할 수 있다. |
| 95 | 　 | 　 | 스코프에 대해서 설명할 수 있다. |

가. 스택 오버플로우가 나도록 함수를 작성해 볼 것. (자바/파이선)

나. 함수 속에서 인수로 전달받은 스트링을 값을 변경하면, 전달해 준 측에서도 그 변수의 값이 변경되는지 알아보는 코드를 작성할 것. (자바/파이선) 결과가 반대로 나오게 코드를 수정해 볼 것.

**메모리**

|  |  |  |  |
| --- | --- | --- | --- |
| 96 | 메모리 | 메모리 종류 | 메모리 종류를 설명할 수 있다. (스택, 힙, 코드 메모리) |
| 97 | 　 | 　 | 인터프리터/컴파일러에서 메모리 설정 방법을 필요에 따라 바꿔 쓸 수 있다. |
| 98 | 　 | 　 | 프로그램의 환경변수로 메모리 사용방식을 정의할 수 있다. |
| 99 | 　 | 　 | 메모리 과다현상 발생시, 추적할 수 있다. |

가. 자바 및 파이선에서 단일 어레이에 최대 몇 개까지 담을 수 있는가? 자기 PC에서 오류가 날 때까지 값을 늘려서, 최대 값을 직접 확인해 볼 것.

나. 리스트에는 최대 몇 개까지 담을 수 있는가? 자기 PC에서 오류가 날 때까지 값을 늘려서, 직접 확인해 볼 것.

다. 자바 또는 파이선의 실행 옵션을 변경해서 해당 최대 치를 늘릴 수 있는 지 알아 볼 것.

라. 메모리 누수가 나도록 프로그램을 짜서 메모리가 누수되는 것을 직접 관찰하도록 할 것. 지속적으로 메모리 그래프가 증가하는 것을 눈으로 확인할 것.

**클래스**

|  |  |  |  |
| --- | --- | --- | --- |
| 100 | 클래스 | OOP | 객체지향의 기본 개념을 설명할 수 있다. |
| 101 | 　 | 　 | 객체와 클래스를 구분하여 설명할 수 있다. |
| 102 | 　 | 　 | 캡슐화를 설명할 수 있다. |
| 103 | 　 | 　 | 상속과 계층구조를 설명할 수 있다. |
| 104 | 　 | 　 | 내부(Private)와 외부공개(Public) 개념을 구분해서 설명할 수 있다. |
| 105 | 　 | 　 | Constant(final), 정적(Static)과 동적 멤버/메소드를 구분해서 설명할 수 있다. |
| 106 | 　 | 　 | 추상 클래스와 인터페이스를 설명할 수 있다. |
| 107 | 　 | 　 | 인터페이스를 클래스와 구분하여 설명할 수 있다. |
| 108 | 　 | 　 | 상속관계와 멤버의 가시성을 설명할 수 있다. |

가. (자바) public/private/final/static member/method 가 골고루 다 들어 있는 자바 클래스를 하나 작성할 것. 각각의 클래스 멤버/메소드에 왜 그런 속성을 부여 했는지 합당하게 설명할 수 있을 것. (단 코드는 완성하지 않아도 되지만, 컴파일해서 동작 하는 것처럼 보이기는 해야 함.)

나. 위에서 만든 자바 클래스를 파이선 클래스로 번역해 볼 것.

**클래스**

|  |  |  |  |
| --- | --- | --- | --- |
| 109 | 　 | 　 | 클래스와 소스코드, 라이브러리의 상관관계를 설명할 수 있다. |
| 110 | 　 | 　 | 객체 생성과 소멸을 설명할 수 있다. |
| 111 | 　 | 　 | 객체 생성 및 메소드 호출에 대한 메모리/성능 이슈를 설명할 수 있다. |
| 112 | 　 | 　 | 가비지 콜렉션을 설명할 수 있다. |
| 113 | 　 | 　 | 중첩 클래스와 익명 클래스를 설명할 수 있다. |
| 114 | 　 | 　 | 객체 복사의 방식을 설명할 수 있다. (DeepCopy) |
| 115 | 　 | 　 | 객체 직렬화 및 저장/전송을 설명할 수 있다. |
| 116 | 　 | 　 | 객체에 대한 JSON/XML/CORBA 표현에 대해서 설명할 수 있다. |

가. 작은 클래스 X를 정의하고, X타입의 멤버를 가진 더 큰 클래스 Y를 정의한다. Y 타입의 클래스를 복사할 때, X 부분이 복사되는 경우와 복사되지 않는 경우를 구분해서 볼 수 있도록 대조적인 코드를 2벌 작성할 것. (Deep Copy) (자바, 파이선)

나. 앞에서 정의한 클래스 Y에 대한 JSON 표현을 표기해 볼 것.

**예외**

|  |  |  |  |
| --- | --- | --- | --- |
| 117 | 예외 | 예외 | 예외를 사용하는 전형적인 패턴을 쓸 수 있다. |
| 118 | 　 | 　 | 예외(Exception)의 프로퍼게이션(Escalation)을 설명할 수 있다. |
| 119 | 　 | 　 | 예외와 함수 스코프를 비교해서 설명할 수 있다. |

가. 수도(Pseudo) 코드 { A; try { B; return C; } catch (Exception e) { D; } finally { E; } return F; } 이 수도(Pseudo) 코드의 A, B, C, D, E, F의 실행 순서가 어떻게 달라질 수 있는 지를 알 수 있는 코드를 작성할 것. (자바, 파이선) C보다 E가 먼저 실행될까? 나중에 실행될까? D와 C 사이에는? D와 E 사이에는?

**코드 디버깅**

|  |  |  |  |
| --- | --- | --- | --- |
| 120 | 　 | 디버깅 | 디버깅용 출력 로깅 코드를 정의할 수 있다. (소스,라인번호) |
| 121 | 　 | 　 | ASSERT의 용도를 설명할 수 있다. |
| 122 | 　 | 　 | 함수의 인자확인(Argument Verification)을 설명할 수 있다. |

가. DEBUG\_PRINT(msg); 라는 함수를 정의할 것. (자바/파이선 각각) 단, msg가 출력될 때, 해당 소스코드 파일명 및 코드 속의 라인 번호가 같이 찍혀야 함.

**에러/경고**

|  |  |  |  |
| --- | --- | --- | --- |
| 123 | 　 | 에러/경고 | 에러 및 경고 메시지를 해석할 수 있다. |
| 124 | 　 | 　 | 에러 및 경고 메시지를 직접 생성해 낼 수 있다. |

가. 자바 및 파이선에서 영어로 출력되는 에러 문장을 하나 샘플로 잡아서 정확한 우리말로 번역할 것. 주어/동사/목적어/보어 등의 문법이 정확히 맞아야 함.

**테스트**

|  |  |  |  |
| --- | --- | --- | --- |
| 125 | TDD | 테스트 | TDD 개발 패턴을 설명할 수 있다. |
| 126 | 　 | 　 | 함수 수준의 TEST 코드를 작성할 수 있다. |
| 127 | 　 | 유닛테스트 | JUnit 테스트 라이브러리를 사용할 수 있다. |

가. 마술 같은 더하기 함수가 있다. 함수명은 add()이다, 값으로 2개의 인수를 받는다. 인수는 String 2개, 리턴 값은 double 1개이다. 이 함수는 “하나”, “스물 다섯”, “23”, “-1”, “Five”, “十六点五”, “2천5백만7천 2백31” 같은 숫자를 입력을 받아서, 의미를 분석한 후에, 수치적으로 정확한 값을 리턴하는 함수이다. 다행히 add()는 우리가 짜지 않아도 된다. 다음 달에 중국에서 작성해서 전달 해 받기로 일정이 잡혀 있는 상태이다. 해당 함수를 공급 받았을 때, 그 함수가 정확하게 동작하는 지를 테스트하는 코드를 작성하라. assert 개념을 이용해서 작성하라. 완전하지는 않더라도, 테스트 코드는 모두 컴파일 되고, 실행 가능해야 한다. (자바/파이선)

**프로파일링**

|  |  |  |  |
| --- | --- | --- | --- |
| 128 | 프로파일링 | 프로파일러 | 프로파일링의 목적과 방법을 설명할 수 있다. |
| 129 | 　 | 　 | 언어별 프로파일링 도구 사용법을 설명할 수 있다. |
| 130 | 　 | 　 | 프로파일링 결과를 해석하고, 병목 지점을 찾아낼 수 있다. |

가. 자기가 지금까지 짰던 아무 코드나 상관없이 하나를 선택해서,
 - 해당 프로그램을 실행하는 도중에, 가장 많이 호출된 메소드(함수)를 알아낼 것.

 - 해당 프로그램에서 가장 많은 CPU 시간을 점유한 함수(메소드)를 찾아낼 것.

**소트**

|  |  |  |  |
| --- | --- | --- | --- |
| 131 | 소트 | 소트 사용 | 소팅의 일반 개념을 설명할 수 있다. |
| 132 | 　 | 　 | 어레이/리스트를 소팅할 수 있다. |
| 133 | 　 | 　 | 비교 연산자를 이용한 일반화된 소팅을 할 수 있다. |
| 134 | 　 | 　 | 해쉬 데이터를 소팅할 수 있다. |
| 135 | 　 | 　 | 소팅의 성능 이슈를 설명할 수 있다. |

가. { { 김길동, 노란색, 30 }, { 빠돌이, 초록색, 27 }, { 이순신, 보라색, 31 }, { Edward, 파란색, 21 }, { 김일성, 주황색, 19 } }
이 데이터를 이름순, 색깔순, 나이 순으로 소트해서 출력할 것. 단 색깔은 무지개색 순서임. (자바, 파이선)

나. 북한 김일성 대학에서 만든 S/W라고 가정해 보자. 미제 단어는 뒤로 빼야 되고, 김일성 이름은 가장 앞으로 보내야 한다. 북한 김일성 대학에서는 만들 법한 이름순 소트 코드를 작성해 보라. (자바/파이선)

**락**

|  |  |  |  |
| --- | --- | --- | --- |
| 136 | LOCK | 락/언락 | 락이 필요한 이유를 설명할 수 있다. |
| 137 | 　 | 　 | 메모리상의 락, 파일의 락을 구분하여 설명할 수 있다. |
| 138 | 　 | 　 | 락의 이용한 패턴인 가드(Guard)를 설명할 수 있다. |
| 139 | 　 | 　 | 락의 이용한 패턴인 싱글톤을 설명할 수 있다. |
| 140 | 　 | 　 | 데드락을 설명할 수 있고, 데드락의 회피 전략을 설명할 수 있다. |

가. 두 개의 스레드가 락을 공유하도록 한 후에, 항상(100%) 데드락이 걸리도록 코드를 짜보라. (자바, 파이선)

나. 위에서 만든 코드에서, 데드락 회피 코드를 작성할 것.

**리눅스**

|  |  |  |  |
| --- | --- | --- | --- |
| 145 | 리눅스 | 리눅스 | 리눅스를 설치할 수 있다. |
| 146 | 　 | 　 | 2개이상의 네트웍 카드에, 네트웍 IP를 세팅할 수 있다. |
| 147 | 　 | 　 | DNS 세팅을 할 수 있다. |
| 148 | 　 | 　 | IPTable을 이용해서 IP를 차단할 수 있다. |
| 149 | 　 | 　 | 라우팅을 지정할 수 있다. |

가. 자기 PC에 가상머신으로 설치된 리눅스의 IP 값을 임의로 변경해 본다. ifconfig 명령어를 이용해서… 변경된 IP 값으로 접속이 되는지 SSH 또는 Telnet으로 접속해 본다.

**가상머신**

|  |  |  |  |
| --- | --- | --- | --- |
| 150 | 가상머신 | 리눅스 | Vbox에 리눅스를 설치할 수 있다. |
| 151 | 　 | 　 | 2개이상의 네트웍 카드에, 네트웍 IP를 세팅할 수 있다. |
| 152 | 　 | 　 | DNS 세팅을 할 수 있다. |
| 153 | 　 | 　 | IPTable을 이용해서 IP를 차단할 수 있다. |
| 154 | 　 | 　 | 라우팅을 지정할 수 있다. |
| 155 | 　 | 　 | Host-Only 네트웍에 연결할 수 있다. |
| 156 | 　 | 　 | 호스팅 머신의 디스크 파티션을 마운트시킬 수 있다. |
| 157 | 　 | 　 | 복수의 Vbox-Linux에 대해서, 서로간, 내외부간 통신을 연결할 수 있다. |
| 158 | 　 | 윈도우 | Vbox에 윈도우(XP)를 설치할 수 있다. |
| 159 | 　 | 　 | 호스팅 머신의 디스크 파티션을 마운팅 시킬 수 있다. |

가. 자기 PC에 설치된 가상머신 Linux에서 가상머신 밖의 DISK 영역을 볼 수 있도록 마운팅 한다. 가상 머신 내에서 해당 마운트 영역에 파일을 생성하고, 가상머신 밖에서 해당 생성된 파일이 읽히는 것을 확인한다.

나. 가상으로 설치된 Window 에서도 똑같이 해 본다.

**OSI 7 레이어**

|  |  |  |  |
| --- | --- | --- | --- |
| 160 | 네트웍 | OSI 7레이어 | 레이어 설명을 할 수 있다. |
| 161 | 　 | 　 | L2, L3, L4, L7을 설명할 수 있다. |
| 162 | 　 | 　 | IP 네트웍의 Ethernet 주소(MAC주소) 및 IP 주소 결정 과정을 설명할 수 있다. |
| 163 | 　 | 　 | 패킷 모니터링을 설명할 수 있다. |
| 164 | 　 | 　 | DNS 조회를 설명할 수 있다. |
| 165 | 　 | L4 | L4 레이어의 주요 이슈를 설명할 수 있다. |
| 166 | 　 | 　 | 포트 포워딩, 로드밸런싱을 설명할 수 있다. |
| 167 | 　 | L7 | L7 레이어의 주요 이슈를 설명할 수 있다. |
| 168 | 　 | 　 | L7 레이어의 주요 서비스를 설명할 수 있다. |
| 169 | 　 | L4 방화벽 | L4 방화벽을 설명할 수 있다. |
| 170 | 　 | L7 방화벽 | L7 방화벽을 설명할 수 있다. |

가. 자기 PC에 설치된 리눅스 가상 머신에서 iptable을 이용해서 포트 포워딩을 한다. 10022 번 포트로 접속을 하면, 22 번 포트(SSH)로 포워딩되도록 세팅한 후에, 10022 번 포트로 접속해서 정상 접속이 됨을 보인다.

**스토리지**

|  |  |  |  |
| --- | --- | --- | --- |
| 171 | 　 | 스토리지 | 스토리지 네트웍을 설명할 수 있다. |
| 172 | 　 | 　 | DAS, SAN형 스토리지를 설명할 수 있다. |
| 173 | 　 | 　 | NAS형 스토리지를 설명하 수 있다. |
| 174 | 　 | 　 | 스토리지 프로토콜을 설명할 수 있다. (iSCISI, …) |

실습없음.

**DBMS SQL 기초**

|  |  |  |  |
| --- | --- | --- | --- |
| 175 | DBMS | SQL 기초 | 테이블, 필드, 키, 색인의 개념을 설명할 수 있다. |
| 176 | 　 | 　 | 테이블 스펙을 작성할 수 있다. |
| 177 | 　 | 　 | ERD(ER-Diagram)을 작성할 수 있다. |
| 178 | 　 | 　 | 테이블의 각종 제약조건을 설명할 수 있다. |
| 179 | 　 | 　 | 여러종류의 색인을 설명할 수 있다. |

가. 어떤 회사의 조직도를 표현하는 DB를 설계하려고 한다. 다음의 설명에 맞는 DB를 설계하라. SQL의 CREATE TABLE, CREATE INDEX 를 이용해서 설계하라. 설계된 결과는 PowerPoint로 ER-Diagram으로 그려보라.

- 사원(user)은 이름, 사원번호, 전화번호, 메일주소 가 있다.
 - 사원 번호가 유니크 키이고, 이름 및 메일주소로 검색이 자주 일어난다.

 - 사원은 부서(dept)에 속하고, 부서에 속할 때, 직급(position)이 정해진다.

 - 한 사원은 2개 이상의 부서에 속할 수 있고, 다른 부서에 속하면 직급도 달라진다. (겸직)

 - 부서는 상위 부서, 하위부서 형태의 트리 구조를 갖는다.

 - 부서는 1년에 한번씩 조직 개편할 때 바뀌는데, 부서 코드, 부서명, 상위부서 코드가 정해진다.

 - 직급은 직급코드, 직급명이 정해진다. 마찬가지로 1년에 한번 정도 직급 체계가 바뀐다.

**SQL**

|  |  |  |  |
| --- | --- | --- | --- |
| 180 | 　 | 　 | Statement와 Prepared Statement를 구분하여 설명할 수 있다. |
| 181 | 　 | 　 | SELECT 성능을 예측할 수 있다. |
| 182 | 　 | 　 | JOIN의 필요성과 방법을 설명할 수 있다. |
| 183 | 　 | 　 | JOIN의 성능을 예측할 수 있다. |
| 184 | 　 | 　 | WHERE, GROUP BY, ORDER BY에 따른 성능 차이를 설명할 수 있다. |
| 185 | 　 | 　 | SQL에 사용되는 전형적인 함수(Functions)들을 설명할 수 있다. |
| 186 | 　 | 　 | SQL을 이용한 부분범위 필요를 설명할 수 있다. |
| 187 | 　 | 　 | SQL Injection을 설명하고 방어책을 설명할 수 있다. |

가. 앞에서 설계한 테이블을 SQLite로 정의하고, SQLite 데이터를 Python/Java로 읽어 내는 것을 해 보자.

 - 샘플로 10명의 데이터와 5개의 부서를 입력하자.

 부장 2명이고, 팀장 4명이고, 개발팀원 중에 1명이 TFT 팀의 팀장으로 겸직하고 있다.

 - 부서는 { 영업부:{ 영업1팀, 영업2팀}, 개발부:{ 개발팀, 기술지원팀}, 신제품TF팀 } 구조이고,

 - 직급은 부장, 팀장, 실장, 사원 이 있다.

나. 앞에서 설계된 테이블을 기준으로
사원명 / 부서명 / 직급명 을 출력하도록 SQL 문을 작성해 보라.

 겸직인 1명이 2번 출력되어야 한다.

다. 같은 리스트를 나이 순으로 출력하도록 코드를 작성하라.

**자바 JDBC**

|  |  |  |  |
| --- | --- | --- | --- |
| 188 | 　 | JDBC | JDBC 기본 패키지 사용을 할 수 있다. |
| 189 | 　 | 　 | JDBC 를 이용한 기본 SELECT를 할 수 있다. |
| 190 | 　 | 　 | JDBC를 이용한 기본 UPDATE를 할 수 있다. |
| 191 | 　 | 　 | JDBC의 커넥션 오픈 및 클로징을 보장할 방법을 설명할 수 있다. |
| 192 | 　 | 　 | JDBC의 커넥션 풀링을 설명할 수 있다. |

라. 앞에서 만든 테이블에 1천 명의 가짜 데이터를 랜덤으로 생성해서 추가해서 넣는다. (자바, 파이선)

마. 앞의 나/다 번 문제를 1천 명이 추가된 상황에서 다시 돌려 본다. 성능의 이슈가 있는 코드를 찾아내서 해결한다.

바. 해당 SQLite DB에 1천명이 아니라 10만명을 넣을 수도 있는가? 100만명은? 천만명은? 최대로 몇 명까지 넣을 수 있는가?

**오라클**

|  |  |  |  |
| --- | --- | --- | --- |
| 193 | 　 | 오라클 | 오라클 서버을 설치할 수 있다. |
| 194 | 　 | 　 | 오라클의 데이터 위치 (파티션 테이블) 등의 정확한 지정을 할 수 있다. |
| 195 | 　 | 　 | 오라클의 사용자 계정을 설정할 수 있다. |
| 196 | 　 | 　 | 오라클의 보안 설정을 할 수 있다. |
| 197 | 　 | 　 | 오라클의 프로세스 및 로그 생성 위치를 설명할 수 있다. |
| 198 | 　 | 　 | 오라클에서 동작하는 SQL을 모니터링할 수 있다. |
| 199 | 　 | 　 | 오라클의 데이터를 주기적으로 백업하는 스크립트를 만들 수 있다. |

실습없음

**MySQL**

|  |  |  |  |
| --- | --- | --- | --- |
| 200 | 　 | MySQL | MySQL 서버을 설치할 수 있다. |
| 201 | 　 | 　 | MySQL의 데이터 위치 (파티션 테이블) 등의 정확한 지정을 할 수 있다. |
| 202 | 　 | 　 | MySQL의 사용자 계정을 설정할 수 있다. |
| 203 | 　 | 　 | MySQL의 보안 설정을 할 수 있다. |
| 204 | 　 | 　 | MySQL의 프로세스 및 로그 생성 위치를 설명할 수 있다. |
| 205 | 　 | 　 | MySQL에서 동작하는 SQL을 모니터링할 수 있다. |
| 206 | 　 | 　 | MySQL의 데이터를 주기적으로 백업하는 스크립트를 만들 수 있다. |

실습없음.

**시간의 표현**

|  |  |  |  |
| --- | --- | --- | --- |
| 207 | 시간 | 시간의 표현 | 컴퓨터 내에서 시간의 다양한 표현을 설명할 수 있다. |
| 208 | 　 | 　 | 각 시간 표현의 장단점과 제약점을 설명할 수 있다. |
| 209 | 　 | 시간 변환 | 각 시간 표현간의 상호간의 변환을 수행할 수 있다. |
| 210 | 　 | 시간측정 | 특정 모듈의 실행시간을 측정할 수 있다. |

가. “2013-12-31 19:31:43” 형태로 표기된 시간을 입력으로 받아서, 정확히 1일 후, 60일 후의 날짜를 같은 포맷으로 출력할 것. (Python, Java)

**개발환경**

|  |  |  |  |
| --- | --- | --- | --- |
| 211 | 개발환경 | 컴파일러 | 컴파일러를 설치할 수 있다. (자바, 파이선, Perl, C++) |
| 212 | 　 | 　 | 컴파일러가 필요로하는 환경세팅을 할 수 있다. |
| 213 | 　 | 　 | 컴파일, 링크를 구분해서 설명할 수 있다. |
| 214 | 　 | 　 | 정적/동적 링킹이 어떻게 다른지 설명할 수 있다. |
| 215 | 　 | 　 | S/W 빌딩 및 패키징을 설명할 수 있다. |
| 216 | 　 | 빌드 | Makefile류의 고전적인 빌드 툴을 만들어 봤다. |
| 217 | 　 | 　 | Makefile의 내부 신택스를 이해한다. |
| 218 | 　 | 　 | Makefile을 이용한 전형적인 빌드 체계를 만들 수 있다. |
| 219 | 　 | 자바/빌드 | Ant를 이용한 빌드를 정의할 수 있다. |
| 220 | 　 | 　 | Ant 빌드스크립트를 수정할 수 있다. |
| 221 | 　 | 이클립스 | 이클립스를 PC에 설치할 수 있다. |
| 222 | 　 | 　 | 이클립스를 이용해서 소스코드를 빌드/컴파일해 봤다. |

별도의 실습 과제 없음.

**SVN/GIT**

|  |  |  |  |
| --- | --- | --- | --- |
| 223 | 　 | SVN/GIT | SVN을 설치하고, 리파지토리를 생성할 수 있다. |
| 224 | 　 | 　 | SVN에 프로젝트를 생성하고, 계정을 생성해 줄 수 있다. |
| 225 | 　 | 　 | SVN에 생성된 프로젝트에 접근하여 다운로드 할 수 있다. |
| 226 | 　 | 　 | SVN에 데이터를 업데이트할 수 있다. |
| 227 | 　 | 　 | SVN의 서브트리를 생성할 수 있다. |
| 228 | 　 | 　 | SVN의 태깅을 설명할 수 있다. |
| 229 | 　 | 　 | SVN의 서브트리 머징에 관련된 문제를 설명하고 해결할 수 있다. |
| 230 | 　 | 　 | 2개 이상의 SVN Client를 사용해 봤다. |
| 231 | 　 | 　 | 코맨드라인으로 SVN checkout, checkin, update, commit을 할 수 있다. |

가. GITHUB에 계정을 만들고, 프로젝트 하나 생성한 후에, 프로젝트 속에 한글->숫자 변환 프로그램을 작성해서 라이브러리로 공개할 것.

 한글->숫자 변환 프로그램은 다음과 같은 변환을 하는 라이브러리임.

 “일” 🡺 1, “오십” 🡺 50, “3만 7천” 🡺 37000, “백만” 🡺 1000000

 코드는 Java 또는 Python중에 한 언어로 구현할 것.

**이중화**

|  |  |  |  |
| --- | --- | --- | --- |
| 232 | 이중화 | 이중화 종류 | 다양한 종류의 이중화를 설명할 수 있다. |
| 233 | 　 | 　 | Active/Standby 이중화를 구성하고 정의할 수 있다. |
| 234 | 　 | 　 | Active/Active 이중화를 구성하고 정의할 수 있다. |
| 235 | 　 | 　 | Active/Backup(Replication) 이중화를 정의할 수 있다. |
| 236 | 　 | 스토리지 | 이중화를 위한 스토리지 구성방법을 설명할 수 있다. |
| 237 | 　 | 서버 | 이중화를 위한 서버 구성 방법을 설명할 수 있다. |
| 238 | 　 | 네트워크 | 이중화를 위한 네트워크 구성 방법을 설명할 수 있다. |
| 239 | 　 | 모니터링 | 이중화 모니터링을 설정할 수 있다. |

별도의 실습 없음.

**FORK/EXEC**

|  |  |  |  |
| --- | --- | --- | --- |
| 240 | FORK/EXEC | 프로세스실행 | 프로세스 실행을 수행할 수 있다. |
| 241 | 　 | 　 | FORK/EXEC/PIPE 구성의 차이점을 설명할 수 있다. |
| 242 | 　 | 　 | 다양한 EXEC를 설명할 수 있다. |
| 243 | 　 | 　 | PIPE로 프로세스들을 연결해서 구성할 수 있다. |
| 244 | 　 | 　 | Child 프로세스의 StdI/O/E를 연결해서 데이터를 주고 받을 수 있다. |

가. 리눅스에서 df -k 명령어의 결과를 읽어서 디스크가 10% 이상 점유된 Partition이 있는지 없는지 알아보는 프로그램을 작성할 것. 10% 이상 점유된 파티션의 개수를 화면에 출력하는 Python 프로그램을 작성한다.

나. Java로 해당 파이선 프로그램을 호출해서, 10% 이상 점유된 파티션의 개수를 숫자로 읽어오도록 프로그램을 구성한다.

**기타**

|  |  |  |  |
| --- | --- | --- | --- |
| 245 | 명령행 | argument | 명령행 인자에 대해서 설명할 수 있다. |
| 246 | 　 | 　 | 명령행 인자를 이용한 옵션 처리를 할 수 있다. |
| 247 | 자바 | System | 패키지 내의 주요 클래스/메소드를 적을 수 있다. |
| 248 | 　 | java.lang.\* | 패키지 내의 주요 클래스/메소드를 적을 수 있다. |
| 249 | 　 | java.util.\* | 패키지 내의 주요 클래스/메소드를 적을 수 있다. |
| 250 | 　 | java.net.\* | 패키지 내의 주요 클래스/메소드를 적을 수 있다. |
| 251 | 　 | java.io.\* | 패키지 내의 주요 클래스/메소드를 적을 수 있다. |
| 252 | 　 | 최적화 | 최적화 옵션 사용법을 안다. |
| 253 | 파이선 | 패키지 | 패키지 구조를 파악하기 위해, dir() 사용법을 안다. |
| 254 | 　 | 기본패키지 | 기본 패키지의 각종 클래스/메소드를 적을 수 있다. |
| 255 | 　 | 최적화 | 최적화 옵션 사용법을 안다. |
| 256 | POSIX | POSIX | POSIX API 함수 명을 나열할 수 있다. |

별도의 실습 없음.