1. 존재 증명이란?
   1) 본인을 증명하기 위한 수단으로 블록체인을 사용하면 좋다.
   2) 블록체인에 데이터를 등록하려면 네트워크에 연결한 후 채굴이 되기를 기다려야 하지만 한 번 등록되면 참조가 빠름
   3) 우리의 시스템에서는?
      => 얻은 명성을 바탕으로 존재를 증명?? 좀 더 생각해 볼 문제

2. 문자열 저장 계약
   1) 블록체인에 저장하는 데이터는 중요하고 데이터 용량이 작게 만들어야 함
   2) 데이터 자체의 용량 크기가 크거나(논문의 내용?) 외부에 노출되더라도 상관없는 데이터는 외부 데이터 베이스를 사용 => 타인이 데이터를 변경하지 않았다는 것을 증명하기 위해 텍스트 파일의 문자열을 해시값으로 블록체인에 저장
   3) Solidity에서 표 형태로 데이터를 저장하기 위해서는 Mapping 함수를 이용하거나 배열을 이용!
   4) 이더리움에서 문자열을 저장하는 경우 숫자 값을 저장할 때보다 Gas가 많이 소모
   5) 과정
      a. 계약 프로그램을 정의
      b. 계약 프로그램 빌드용 Data 부분을 출력
         => solc -o [출력할 곳] --bin --optimize [컴파일할 Solidity 소스 파일]
            ex) solc -o ./ --bin --optimize KeyValueStore.sol
      c. 계약 정보를 가져온다.
         => solc --abi [컴파일할 Solidity 소스 파일]
      d. geth를 시작한다.
         => geth --networkid 4649 --nodiscover --maxpeers 0 --datadir [위치] console 2>> [geth.log 위치]
      e. 계약 등록자의 계정 잠금을 해제한다.
         => personal.unlockAccount(eth.accounts[0], “패스워드", 0)
      f. 계약을 블록체인에 등록한다.
         => keyvaluestoreContract(임의의 변수) = web3.eth.contract(c에서 가져온 ABI 정보)
            keyvaluestore = keyvaluestoreContract.new({실행할 계정, data:’0x[b에서 획득한 데이터]’, gas:3000000})
      ex) keyvaluestoreContract = web3.eth.contract([{“constant":true,"inputs":[{"name":"_key","type":"uint256"}],"name":"getValue1","outputs":[{"name":"","type":"string"}],"payable":false,"stateMutability":"view","type":"function"},{“constant":true,"inputs":[{"name":"_key","type":"uint256"}],"name":"getValue2","outputs":[{“name":"","type":"string"}],"payable":false,"stateMutability":"view","type":"function"},{“constant":true,"inputs":[{"name":"_value1","type":"string"},{“name":"_value2","type":"string"}],"name":"setValue","outputs":[{“name”:"","type":"uint256"}],"payable":false,"stateMutability":"view","type":"function"}])
      g. 채굴을 실시한다.
      h. 계약이 블록체인에 등록됐는지 확인한다. 트랜잭션이 실행되었다면 블록 번호가 표시!!
         => eth.getTransaction(f에서 획득한 트랜잭션 결과 해시 값)
      i. 계약에 접근하기 위한 변수를 정의한다.
         => contractObj = eth.contract(keyvaluestore.abi).at(keyvaluestore.address)
      j. 블록체인의 계약에 값을 등록하기 위해서는 다음과 같이 수행한다.
         => contractObj.[등록함수].sendTransaction(등록 함수의 인수, 실행 계정)
      k. 블록체인의 값을 참조하기 위해서는 다음과 같이 수행한다.
         => contractObj.[참조함수].call(참조 함수의 인수, 실행 계정)

3. 계약 생성 관련
    1) 개인정보 취급같은 경우 공용 블록체인에 개인정보를 기록해두면 해당 블록체인 노드에 참가하고 있는 사용자가 개인정보를 열람할 수가 있다….! 따라서 이 문제에 대한 해결책이 필요
        a. 개인정보에 별도의 패스워드 설정 => 아직 연구가 충분치 않음
        b. 외부데이터베이스를 이용! => 개인정보 등은 RDBMS에 저장하고, 트랜잭션 데이터는 블록체인에!
    2) 확인 처리 내용이 바뀌지 않는다면 블록체인에서 처리, 웹 응용 프로그램에서는 확인 처리 내용의 변경과 상관 없이 모든 확인 처리를 수행
    3) 클라이언트 응용 프로그램과 이더리움간 처리를 위한 API를 만들 것
    4) Gas를 설정해두는 게 좋음(미사용분의 Gas는 반환)

4. 본인 확인 서비스
    1) 계약을 등록
    2) 인증 조직 정보를 등록, 입학, 취업 사실을 등록
    3) 본인 정보를 등록, 경력을 볼 사람에게 열람권 부여
    4) 기간 내와 기간 종료 후 A의 본인 확인 정보를 열람
    5) public을 선언하면 제한 없이 열람할 수 있음
    6) A의 주소에는 다음과 같은 정보가 들어가게 된다.
        a. 참조 허가 여부(열람을 허가할지 말지)
        b. 승인 시 블록 번호
        c. 열람 기간을 종료할 블록 번호
        d. 열람을 허용할 주소(기업의 주소?)