Assignment1: 블록체인 작업 증명 (Proof-of-Work) 시뮬레이션 보고서

1. 개요 (Introduction)

본 보고서는 과제 1에서 요구하는 블록체인 작업 증명 (PoW) 시뮬레이션 프로그램 (`generate_blocks.py`)와 블록 유효성 검사 프로그램 (`verify_block.py`)의 구현 내용과 실행 요구 사항 및 예상 결과를 설명합니다.목표는 SHA256 해시 알고리즘을 사용하여 10개의 블록을 순차적으로 연결하고, 각 블록의 해시값이 2^240미만이 되도록 하는 Nonce를 찾는 것입니다.

2. 프로그램 실행 요구 사항 (Execution Requirements)

2.1 개발 환경 및 종속성

항목	요구사항	비고
언어	Python 3.x	Python 인터프리터 설치 필수
라이브러리	`hashlib` , `json` , `time` , `sys` , `re`	모두 Python 표준 라이브러리이므로 별도의 설치 불필요

2.2 실행 방법

두 프로그램 모두 명령줄(Command Line)에서 실행해야 합니다. 두 파일 (`generate_blocks.py` , `verify_block.py`)이 같은 폴더에 있어야 합니다.

1. 블록 생성

```
python generate_block
```

2. 블록 검증 (예시 : Block1.txt 검증)

```
python verify_block.py Block1.txt
```

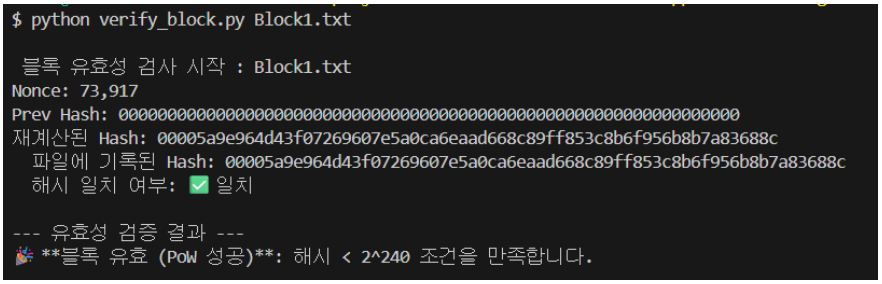
3. 프로그램 예상 결과 및 검증 기준 (Expected Outcome)

3.1 블록 생성 프로그램 (`generate_blocks.py`) 예상 결과

- 실행 로그 : 1번부터 10번까지 각 블록에 대해 채굴 시작, Nonce 값, 최종 해시 값, 소요 시간 그리고 '파일 저장 완료' 메시지를 순차적으로 출력합니다.
- 결과 파일 : 현재 실행 폴더에 `Block1.txt` 부터 `Block10.txt` 까지 10개의 텍스트 파일이 생성됩니다.
- 블록체인 연결 : 각 `BlockN.txt` 파일의 `Prev` 해시는 `Block(N-1).txt` 의 `Final Hash` 와 정확히 일치해야 합니다.
- 난이도 조건 충족 : 모든 블록의 `Final Hash` 는 2^240 미만의 값을 가집니다. 이는 16진수 해시 문자열이 최소 4개 이상의 선행 '0'으로 시작함을 의미합니다.

3.2 블록 유효성 검사 프로그램 (`verify_block.py`) 예상 결과

- 프로세스 : 입력된 `BlockN.txt` 파일을 파싱하여 `Block` , `Nonce` , `Tx` , `Prev` 데이터를 추출한 후 이 정보를 기반으로 SHA256 해시를 재계산합니다.
- 출력 메시지
    1. 재계산된 해시와 파일에 기록된 `Final Hash` 가 일치함을 확인합니다.
    2. 재계산된 해시 값의 난이도가 조건을 만족하면 블록 유효 메시지를 출력합니다.



4. 결론 (Conclusion)

본 과제에서 구현한 두 프로그램은 SHA256 PoW 시뮬레이션의 모든 요구사항을 충족하며, 생성된 블록은 유효성 검사 프로그램을 통해 그 무결성(Integrity)이 성공적으로 확인될 수 있습니다.