Rebase 집중 실습1:

가장 오래된 역사 부터 두번째 commit 이후에 새로운 커밋 commit 3개 넣기

실습1: 가장 오래된 역사 부터 두번째 commit 이후에 새로운 커밋 commit 3개 넣기

* 참고: 지난시간에 fork 받아서 clone했던 git-training 재활용!

```
$ cd git-training
$ git rebase -i --root
$ touch hello_1.c
$ git add hello_1.c
$ git commit -m "test: add hello_1.c"

$ touch hello_2.c
$ git add hello_2.c
$ git commit -m "test: add hello_2.c"

$ touch hello_3.c
$ git add hello_3.c
$ git commit -m "test: add hello_3.c"

$ git rebase --continue
```

실습1: 가장 오래된 역사 부터 두번째 commit 이후에 새로운 커밋 commit 3개 넣기

* 결과 확인하기

(새로운 commit 3개)

Rebase 집중 실습2:

위 예제에서 넣은 (중간에 낀) 3개를 commit을 1개로 합치기

실습2: 위 예제에서 넣은 (중간에 낀) 3개 commit을 1개로 합치기

* 참고: hello_3.c 커밋, hello_2.c 커밋, hello_1.c 커밋 총 3개의 커밋을 합친다.

(1) hello_3.c 커밋과 hello_2.c 커밋을 합친다.

```
$ git rebase -i --root
$ git reset --soft HEAD~1
$ git status

$ git commit --amend
$ git rebase --continue
```

* reset --soft HEAD~1 설명:

--hard와는 다르게 commit 정보만 삭제하고 파일 변경분은 남겨둔다.

git reset --hard HEAD~1 은 최신커밋삭제 뿐만아니라 파일의 변경분도 완전히 삭제한다.

실습2: 위 예제에서 넣은 (중간에 낀) 3개 commit을 1개로 합치기

* 참고: hello_3.c 커밋, hello_2.c 커밋, hello_1.c 커밋 총 3개의 커밋을 합친다.

(2) hello_3.c 커밋과 hello_2.c 커밋을 합쳐진 결과를 확인한다.

실습2: 위 예제에서 넣은 (중간에 낀) 3개 commit을 1개로 합치기

* 참고: hello_3.c 커밋, hello_2.c 커밋, hello_1.c 커밋 총 3개의 커밋을 합친다.

(3) 이전 커밋(helo_2.c + hello_3.c)과 hello_1.c 커밋 합치기

```
$ git rebase -i --root
$ git reset --soft HEAD~1
$ git status

$ git commit --amend
$ git rebase --continue
```

실습2: 위 예제에서 넣은 (중간에 낀) 3개 commit을 1개로 합치기

* 참고: hello_3.c 커밋, hello_2.c 커밋, hello_1.c 커밋 총 3개의 커밋을 합친다.

(4) 이전 커밋(helo_2.c + hello_3.c)과 hello_1.c 커밋 합치기

Rebase 집중 실습3:

가장 오래된 역사에서 두번째 커밋

"Add knapsack problem PDF" 삭제하기

실습3: 가장 오래된 역사에서 두번째 커밋 "Add knapsack problem PDF" 삭제하기

(1) 가장 오래된 커밋에서 두번째 커밋을 찾아내고 지운다.

```
$ git rebase -i --root
$ git status

$ git reset --hard HEAD~1
$ git rebase --continue
```

실습3: 가장 오래된 역사에서 두번째 커밋 "Add knapsack problem PDF" 삭제하기

(2) 가장 오래된 커밋에서 두번째 커밋을 지운결과를 확인한다.

git blame 명령:

해당 소스라인 대해서 누가 마지막으로 수정을 했는지

commit ID 추적이 가능하다

실습: Node.js 소스코드 git clone 하고 간단하게 git blame 테스트

```
f2282bb8128 src/node.cc (Sam Roberts          2017-02-20 06:18:43 -0800  758)
29a71bae40f src/node.cc (Anna Henningsen      2018-08-10 02:45:28 +0200  759)   std::vector<char*> v8_args_as_char_ptr(v8_args
29a71bae40f src/node.cc (Anna Henningsen      2018-08-10 02:45:28 +0200  760)   if (v8_args.size() > 0) {
29a71bae40f src/node.cc (Anna Henningsen      2018-08-10 02:45:28 +0200  761)     for (size_t i = 0; i < v8_args.size(); ++i)
29a71bae40f src/node.cc (Anna Henningsen      2018-08-10 02:45:28 +0200  762)       v8_args_as_char_ptr[i] = &v8_args[i][0];
29a71bae40f src/node.cc (Anna Henningsen      2018-08-10 02:45:28 +0200  763)     int argc = v8_args.size();
29a71bae40f src/node.cc (Anna Henningsen      2018-08-10 02:45:28 +0200  764)     V8::SetFlagsFromCommandLine(&argc, &v8_args_
29a71bae40f src/node.cc (Anna Henningsen      2018-08-10 02:45:28 +0200  765)     v8_args_as_char_ptr.resize(argc);
29a71bae40f src/node.cc (Anna Henningsen      2018-08-10 02:45:28 +0200  766)   }
f2282bb8128 src/node.cc (Sam Roberts          2017-02-20 06:18:43 -0800  767)
f2282bb8128 src/node.cc (Sam Roberts          2017-02-20 06:18:43 -0800  768)   // Anything that's still in v8_argv is not a V
eb664c3b6df src/node.cc (Anna Henningsen      2019-01-06 22:55:09 +0100  769)   for (size_t i = 1; i < v8_args_as_char_ptr.siz
eb664c3b6df src/node.cc (Anna Henningsen      2019-01-06 22:55:09 +0100  770)     errors->push_back("bad option: " + std::stri
f2282bb8128 src/node.cc (Sam Roberts          2017-02-20 06:18:43 -0800  771)
eb664c3b6df src/node.cc (Anna Henningsen      2019-01-06 22:55:09 +0100  772)   if (v8_args_as_char_ptr.size() > 1) return 9;
f2282bb8128 src/node.cc (Sam Roberts          2017-02-20 06:18:43 -0800  773)
eb664c3b6df src/node.cc (Anna Henningsen      2019-01-06 22:55:09 +0100  774)   return 0;
eb664c3b6df src/node.cc (Anna Henningsen      2019-01-06 22:55:09 +0100  775) }
```

Blame 실습1:

node_http_parser.cc 의 Parser 클래스 만든

최초 Commit 을 찾아내라

실습1: node_http_parser.cc 의 Parser 클래스 생성 commit 찾기

```
# Node.js 소스폴더(node)에서 node_http_parser.cc 가 있는 해당 경로로 이동하자
$ cd src/;

# blame 을 통해서 Parser 클래스 소스구현 라인중에 최초 commit 을 찾아보자
$ git blame node_http_parser.cc;

# 찾은 commit이 Parser 클래스 최초구현 commit인지 확인
$ git show <commit ID>

# 힌트: 특정 commit 이전 역사로 되돌리기
$ git reset --hard <commit ID>~1
```

Blame 실습2:

node.cc 파일이 생성된

진짜 최초 Commit 을 찾아내라

실습2: node.cc 파일이 생성된 진짜 최초 commit 찾기

```
# Node.js 소스폴더(node)에서 node.cc 가 있는 해당 경로로 이동하자
$ cd src/;

# git log 을 통해서 node.cc 파일이 생성된 최초 commit 을 찾아보자
$ git log --reverse -- node.cc;

# 만약 최초 생성 commit 이 아니라 단순 경로 이동이라면
# 그 commit ID 를 바로 직전 commit 의 시점으로 돌아가보자
$ git reset --hard <commit ID>~1

# src/ 폴더아래가 아닌 node/ 아래로 이동해서
$ cd ../

# 다시한번 log — reverse 로 추적
$ git log --reverse -- node.cc;
```

Blame 실습3:

node 프로젝트가 처음 만들어질때

최초부터 Commit 3 개 찾아내라

실습3: node 프로젝트의 최초 commit 부터 3개 찾고 확인하기

```
# Node.js 폴더 아래에서
$ git log --oneline --reverse | head -3

# 최초 commit 3가지 git show 로 확인하기
$ git show <commit ID>
```