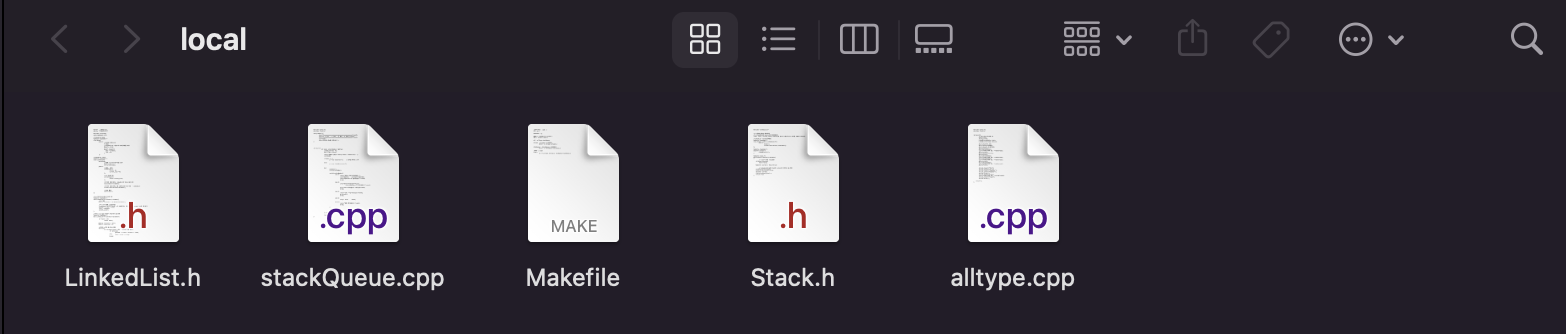
5주차 결과보고서

전공: 수학/컴퓨터공학 학년: 3학년 학번: 20181288 이름: 윤성호

1. Git init 명령어를 이용하여 로컬 저장소를 생성한다. 현재 디렉토리 이름은 local로 설정하였고 내부 파일은 4주차 실습에서 사용한 파일을 넣어주었다. [사진 1]은 디렉토리 내부의 파일들을 캡처한 것이고, [사진 2]는 git init 명령어를 이용하여 git repository를 초기화한 화면이다.



[사진 1]

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 2]

git add \* 를 이용하여 파일들을 local 디렉토리 안의 모든 파일들을 git repository에 추가해준다. 이는 git add . 명령어와 같은 역할을 한다. 그 이후 git remote add origin <https://github.com/SoftCoffee1/comsil_5_github.git> 라는 명령어를 입력하여 로컬저장소와 원격저장소를 연결해준다. 이 때 원격저장소의 이름을 매번 쓰기는 번거로우니 origin 이라는 이름으로 원격 저장소 불러올 수 있도록 해주었다. 그 다음, git commit -m “실습 1”을 통해 변경된 내용을 저장소에 새로운 버전으로 확정시킨다. 이때 커밋의 설명은 “실습 1” 로 하였다. 마지막으로 git push origin master 명령어를 입력하여 원격 저장소에 반영해준다. origin의 master branch에 반영해준다는 의미이다. [사진 3]은 local 이라는 로컬 저장소를 깃허브 원격저장소에 반영해주는 과정을 보여준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

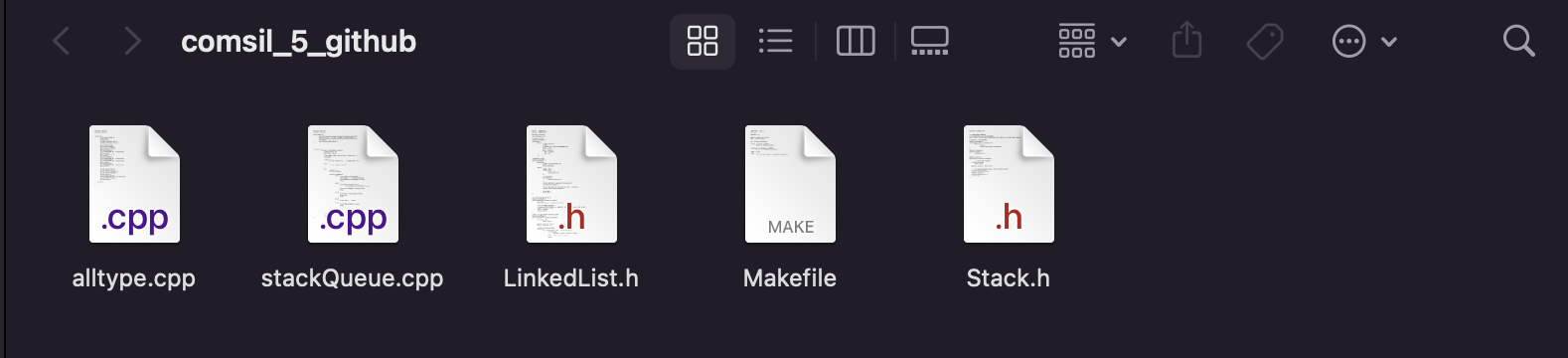
[사진 3]

위의 과정에 의해 터미널상에서 github에 파일을 업로드하는 과정을 진행하였다. 이제, 원격 저장소에 있는 파일을 로컬 저장소로 받아오는 과정을 설명하려고 한다. 현재 폴더에 받아오기 위해서는 git clone <원격저장소 주소> 를 입력하면 된다. 이를 터미널에서 한 과정이 [사진 4] 이고 [사진 5]를 통해 받아온 폴더 내부를 살펴보며 잘 받아온 것인지 확인한다. 받아온 폴더의 이름은 원격 저장소의 이름과 같은 comsil\_5\_github으로 정해진다.

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 4]



[사진 5]

이 후, 받아온 comsil\_5\_github애 파일을 추가하여 다시 원격저장소에 반영하는 작업을 진행했다. 이때 파일 추가는 touch a.txt 명령어를 통해 텍스트 파일 하나를 생성하였고, git add \*를 통해 모든 파일을 추가해주었고, git commit -m “실습 2” 를 통해 이전의 “실습 1” 설명과는 다른 커밋이라는 것을 내용설명에 추가해주었다. 이후, git push origin master를 통해 원격 저장소에 성공적으로 로컬 저장소의 컨텐츠를 반영할 수 있었다. 이 과정은 [사진 6]을 통해 확인할 수 있다.   
텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 6]

1. 생성된 원격 저장소의 주소는 아래와 같다.

<https://github.com/SoftCoffee1/comsil_5_github>

1. 혼자 프로젝트를 진행할 때는 매번 로컬 저장소에 있는 코드가 가장 최신 코드라는 것이 보장된다. 그 위에 어떠한 수정을 하더라도 가장 최신버전에서 수정을 하는 것이기에 문제가 발생하지 않는다. 그러나 규모가 큰 프로젝트를 진행하게 되면 여러 사람이 협업을 하며 작업을 해야하는 상황이 생긴다. 이럴 때는, 각자의 노트북에 저장되어있는 코드가 가장 최신 버전의 코드라고 단정지을 수 없다. 매번 수정사항이 생길때마다 서로에게 연락을 주어 가장 최신버전의 코드가 무엇인지 소통을 통해서만 알 수가 있다. 또한, 협업자와 내가 동시에 코드를 수정한 상황에서는 두 코드를 적젏히 합쳐야 하는데, 이 과정은 프로젝트의 사이즈가 커질수록 어려워지며, 불가능할 수도 있다. 계속해서 업데이트 되는 상황에서 무엇이 최신이고, 그 이전 버전과 어떠한 차이가 생겼고, 현재 버전에서 여러가지의 수정사항이 나올 경우, 결과적으로 어떠한 수정사항을 최종 수정사항으로 결정할 것인지의 문제들을 github을 사용하면 모두 해결이 가능하다. 협업자들이 코드를 수정한 후 github에 올리면 가장 최신의 코드가 무엇인지 바로 확인이 가능하고, commit을 한 후 코드를 반영하기 때문에 이전 버전의 코드가 궁금한 경우에도 돌아가기가 매우 용이하며, 현재 버전에서 여러가지 수정사항이 나올 경우 branch를 뻗어나가면서 다른 협업자들과의 코드와 충돌이 없도록 작업할 수 도 있다. 이러한 수정사항을 merge하여 하나의 branch로 다시 통합할 수도 있다. 이러한 작업을 특정 환경에서가 아닌 웹 상에서 할 수 있다는 것은 노트북만 있으면 어디서든 원활한 협업이 가능하게 된다는 것을 의미하기도 한다. 이러한 이유때문에 개발자들이 github를 애용하는 것이라고 생각한다.