그런즉 너희가 먹든지 마시든지 무엇을 하든지 다 하나님의 영광을 위하여 하라 (고전 10:31)

NOTE: The following materials have been compiled and adapted from the numerous sources including my own. Please help me to keep this tutorial up-to-date by reporting any issues or questions. Send any comments or criticisms to idebtor@gmail.com Your assistances and comments will be appreciated.

Lab 5 - make/makefile 도구를 사용하여 빌드 프로세스 실습

목적

1. `makefile` 을 다양한 형태로 작성
2. 라이브러리 파일 만들어 보기 – `libsort.a`

Getting Started

- 제공 되는 파일
    - `sortDriver.cpp, comparator.cpp, printlist.cpp` – 코딩이 완료된 소스 파일
    - `bubble.cpp, insertion.cpp` – 코딩이 완료된 소스 파일
    - `selection.cpp quick.cpp merge.cpp` – 부분적으로 코딩된 소스 파일 (뼈대 코드), 함수포인터를 사용하여 **내림차순으로도 작동되도록 코딩해야 할 파일.**
    - `UsingStaticLib.md` - 참조용

Step 1. “Build Process” 강의 따라하기

참조를 위해 5 개의 정렬 알고리즘 파일들(`bubble.cpp, insertion.cpp, selection.cpp merge.cpp, quick.cpp`)과 helper 파일들(`sortDriver.cpp, printList.cpp, comparator.cpp`) 및 `makefile` 파일들이 제공됩니다.

- 먼저 ”Build Process” 영상을 참고하여 파일을 작성하고 빌드를 시도합니다. 그리고,
- 아래에 `makefile` 들을 생성하여 작동되는지 테스트합니다.
    - `makefile`
    - `make.1`
    - `make.2`
    - `make.3`
    - `make.libsort`
- **과제를 하면서 정렬 알고리즘 파일들(`bubble.cpp, insertion.cpp, selection.cpp, merge.cpp, quick.cpp`)에 존재하는 `#if 1/0` 명령들을 빌드하는 단계에 따라 `on/off` 조정한다.**
- **일단, sortDriver.cpp 와 필요한 파일들과 함께 실행파일을 만들고 실행해봅니다.** 이 때 build 를 위해 사용하는 파일이 make.3 파일입니다. 이 때 완성된 실행파일은 내림차순으로 정렬하는 부분을 다 완성하지 않은 단계일 것입니다.

```
make -f make.3
./sortDriver
```

Step 2. Selectionsort, Mergesort & Quicksort 알고리즘 완성하기

- 부분적으로 코딩된 소스 파일 (뼈대 코드)에 있는 알고리즘들이 함수 포인터를 사용하여 **내림차순으로도 작동되도록 작성합니다.**
- 이미 완성되어 제공된 bubblesort 및 insertionsort 알고리즘에서 사용한 비교 함수 포인터를 인수로 추가하는 코드를 살펴보십시오.
- 그리고, 나머지 정렬 알고리즘들(selection, quick, merge)에서도 유사한 방법으로 비교 함수 포인터를 사용하도록 코드를 수정할 때, 코딩하기 쉬운 selectionsort 함수부터 시작하십시오.
    - Selection.cpp 파일에서 main()함수를 활성화하여 집중적으로 코딩하면서 테스팅 해볼 수 있습니다.

```
g++ selection.cpp -o selection
./selection
```

    - 그 다음 같은 방법으로 quicksort 를 도전하십시오. 우리가 여기서 하고자 하는 것은 알고리즘에서 비교하는 부분들을 주의 깊게 비교 함수 포인터로 대체하는 것입니다.

```
g++ quick.cpp -o quick
./quick
```

  - Mergesort 를 구현하는 것은 까다로우니, 가장 나중에 도전하는 것이 좋습니다.

```
g++ merge.cpp -o merge
./merge
```

- Selectiosort, quicksort, mergesort 가 모두 완성되어 내림차순으로 정렬되는 것을 확인한 후에는 각 파일에 있는 main() 부분을 비활성화 합니다.

Step 3. 모든 정렬 알고리즘 테스트 하기

- 모든 정렬 파일들(bubble.cpp, insertion.cpp, selection.cpp, quick.cpp, merge.cpp)에 있는 main() 함수가 비활성화 된 상태에서, 다시 sortDriver 와 함께 build 를 해서 테스트하면 됩니다. 이 때 사용하는 빌드 파일은 make.3 파일입니다.

```
make -f make.3
./sortDriver
```

Step 4. libsort.a 만들기

- Main()함수가 모두 비활성되어 만든 정렬 파일들(bubble.cpp, insertion.cpp, selection.cpp, quick.cpp, merge.cpp)들의 ~.o 오브젝트 파일들을 libsort.a 를 만드십시오. 이 때 사용하는 build 파일은 make.libsort 가 되어야 합니다.

Step 5 libsort.a 를 링크하여 sortDriver 실행파일 만들기

- 이제는 다음 한 줄의 코맨드로 sortDriver 실행파일이 만들어져야 합니다.

```
g++ sortDriver.cpp -o sortDriver  -L./  -lsort -I../../include
```

제출파일 목록

```
아래에 파일들을 제출한다. 작동되지 않는 스크립트를 제출할 경우 페널티 존재.
 - make.3                    # build sortDriver.cpp
 - make.libsort              # libsort.a 빌드하기 위한 스크립트
 - selection.cpp, quick.cpp, merge.cpp  # 내림차순 작동해야 함
```

```
        - libsort.a                 # windows 사용자 제출파일
        - libsort_mac.a             # mac 사용자 제출파일
```

```
      Happy Coding~~
```

One thing I know, I was blind but now I see. John 9:25