

## Lab 13 (6/08) Quick Sort

- arranged by TA Baesoon Kim

**Due : ~2023.06.14(Wed) 23:59, Late Submission : ~2023.06.15(Thu) 23:59**

- Quick Sort Implementation.
- 아래의 사진과 같이 [input].txt 파일을 입력받아 Quick sort을 수행하고 결과값을 [output].txt 파일에 출력하여 저장한다.
- Implement the Quick Sort algorithm and run sorting by receiving the [input].txt file as input commands as below, and the output result is stored in the [output].txt file.

```
Open ▾ inpu... ~/Do... Save ≡ - □ × Open ▾ out... ~/Do... Save ≡ - □ ×
```

```
1 10
2 8 1 4 9 0 3 5 2 7 6
```

```
1 left : 0, right : 9, pivot : 6
2 8 1 4 9 0 3 5 2 7 6 |
3 2 1 4 9 0 3 5 8 7 6
4 2 1 4 5 0 3 9 8 7 6
5 2 1 4 5 0 3 6 8 7 9
6 left : 0, right : 5, pivot : 3
7 2 1 4 5 0 3 - - - -
8 2 1 0 5 4 3 - - - -
9 2 1 0 3 4 5 - - - -
10 left : 0, right : 2, pivot : 0
11 2 1 0 - - - - - -
12 0 1 2 - - - - - -
13 left : 1, right : 2, pivot : 2
14 - 1 2 - - - - - -
15 left : 4, right : 5, pivot : 5
16 - - - - 4 5 - - - -
17 left : 7, right : 9, pivot : 9
18 - - - - - - 8 7 9
19 left : 7, right : 8, pivot : 7
20 - - - - - - 8 7 -
21 - - - - - - 7 8 -
22
23 sorting result :
24 0 1 2 3 4 5 6 7 8 9
```

```
Tab Width: 8 ▾ Ln 2, Col 20 ▾ INS Tab Width: 8 ▾ Ln 2, Col 31 ▾ INS
```

- <input> :
  - 첫째줄 : Sorting 할 숫자의 개수, 둘째줄 : Sorting 할 숫자의 sequence
    - Sorting 할 숫자의 개수는 최대 100.
  - 둘째줄 : 숫자 Sequence의 범위는 0 ~ 99 , 중복된 값 입력 허용.
  - 1st line : the number of numbers to be sorted, 2nd : sequence of numbers to be sorted
  - the number of numbers to be sorted is less or equal to 100.
  - integer numbers in the sequence are in the range of 0 ~ 99 , duplicated numbers allowed.
- <output> :

- 현재 **Array** 구간 정보(**Left, Right, Pivot Value**의 **index**) 및 해당구간의 **첫번째 상태값(array value)** 출력 후, 구간 내에서 **Swap**이 일어날 때마다 **상태값(array value)** 출력
- 구간의 값 및 **Sorting Result** 출력시 “%-3s”, “%-3d” 를 사용하여 출력되는 간격을 숫자 혹은 “-”포함 3칸으로 고정.
- **Left**는 구간의 첫 인덱스를 출력, **Right**는 구간의 마지막 인덱스를 출력.
- **Pivot**은 **Pivot**으로 정해진 숫자의 값 출력.
- **Pivot**은 항상 정해진 구간의 마지막 인덱스에 위치한 숫자로 선정.(가장 오른쪽 숫자)
- information of present state of array section(**Left, Right, Pivot Index**) and the initial state(the array value) of the section should be printed, and after that, whenever the swap happens, the array value should be printed.
- when printing key values in the section and the sorting result, use “%-3s”, “%-3d” to fix the alignment space in 3 spaces including digits or “-”.
- Left should be printed as the first index of the section, and Right should be printed as the last index of the section.
- Pivot should be printed as the value of the chosen Pivot.
- The number placed in the last index of the section is always chosen as a pivot(the right-most number)

#### <CreateArray> :

- **Array**를 생성하는 함수
- **input** : input text file에서 받은 첫번째 줄의 값 (**size**)
- **output** : 생성한 **Array**의 포인터
- **Function to create Array**
- **Input** : the value of the first line received from the input txt file (size)
- **Output** : Pointer to the created Array

#### <QuickSort> :

- **Quick sort**를 진행하는 함수
- 재귀적인 형태로 구현
- **Function to execute quick sort**
- **Implement in recursive form**

#### <Partition> :

- **Array**를 2개의 **sub-list**로 나누는 함수
- 가장 오른쪽 **data**를 **pivot**으로 선정
- **Left, right, pivot**인덱스를 처음에 출력
- **Swap**을 할 때마다 **array**의 **value**들을 출력 (**PrintArray**함수에 구현)
- **Index Print format** :  
  - “**left : %d, right : %d, pivot : %d\n**”
- **A function that divides an Array into 2 sub-lists**
- **Select the rightmost data as the pivot**
- **Left, right, pivot indices should be printed at first**
- **Whenever Swap is performed, print the values of the array (implemented in the PrintArray function)**

- Index Print format :

“left : %d, right : %d, pivot : %d\n”

<PrintArray> :

- Array value들을 출력하는 함수.
- Print format :
  - “%-3s” : 범위를 벗어난 array value들은 -로 채워서 출력
  - “%-3d” : 범위 내에 있는 array value들의 값을 해당 형태로 출력
- A function that prints array values.
- Print format:
  - “%-3s” : Out of range array values are filled with -
  - “%-3d” : Print the values of array values within the range

<DeleteArray> :

- Array의 메모리를 해제합니다.
- Deallocate Array's memory.

<Structure & Function Format>

## Structure

```
typedef struct Array Array;
struct Array{
    int size;
    int* values;
};
```

## Function

```
Array* CreateArray(int size);
void QuickSort(Array* array, int left, int right);
int Partition(Array* array, int left, int right);
void PrintArray(Array* array, int left, int right);
void DeleteArray(Array* array);
```

- 위 사진과 같은 Struct 구조체를 사용하셔야 합니다.
- 위 사진과 같은 함수들을 형식에 맞게 구현해주시면 됩니다.
- Struct format above should be used for implementation
- Functions should be implemented in appropriate format as above.

<File Name Format>

- [StudentID].c     ex) 20XXXXXXXXX.c

<Execution>

- gcc 20XXXXXXXXX.c -o 20XXXXXXXXX
- ./20XXXXXXXXX [input\_file\_name] [output\_file\_name]
- !!! 꼭 제공되는 testcase로 실행시켜보시기 바랍니다!!!!!!,
- !!! Run your solution code with the provided test case above and check whether it works properly !!!

<Issue>

- 코드 작성시 주석을 적어주시기 바랍니다. 주석이 없는경우 Cheating으로 간주될 수 있습니다.
- 제공된 testcase는 채점 case에 포함됩니다. 모두 알맞게 나오는지 확인해보시기 바랍니다.
- 파일 입출력은 argv[] 를 사용하여 구현해주시기 바랍니다.
- 제출 마감 시간 이전의 가장 최신 버전의 commit을 기준으로 채점할 예정입니다.
- 제출 파일과, 폴더 naming 은 꼭 지정된 형식으로 해주셔야 합니다.

- 출력시 꼭 정해진 형식에 맞게 출력해주셔야 합니다. 모든 출력 메시지의 알파벳은 소문자만 사용하여 출력 합니다.
- All the messages must be printed out according to the appropriate format as shown in the example above. Only lowercase letters should be used for the alphabets in output messages.

## <Directory Format>

- 아래와 같이 git 프로젝트 폴더에 "lab13" 폴더 생성후, "lab13" 폴더 안에 "20XXXXXXXXX.c" 파일을 위치시키시면 됩니다.
- After creating the "lab13" directory in the git-project-directory as below, place the "20XXXXXXXXXX.c" file in the "lab13" directory.

[illegible]

2023\_CSE2010\_20XXXXXXXX/ (GitLab project directory)

---

---lab11/

-----2020XXXXXX.c

\_\_\_\_\_

---lab12/

-----2020XXXXXX.c

\_\_\_\_\_

---lab13/

-----2020XXXXXX.c

[illegible]