1 번 문제

1. (Programming: **35 points**)
The selection sort is known to produce unstable sorting results. The attached code is a simple instance of unstable sorting results.
First, explain the stable sorting result of the following input, and then implement '`selection_sort_stable`' that produces the stable sorting result.

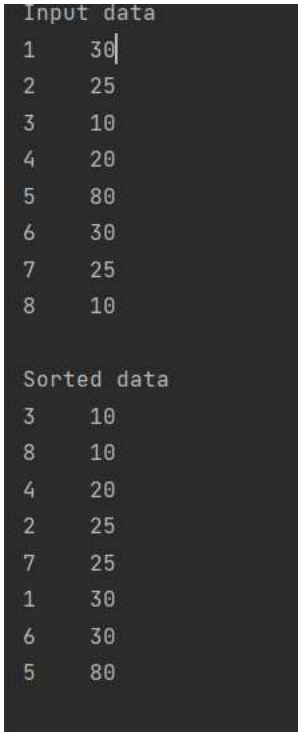
Note) It is not mandatory to follow the data structure used in the attached code. Feel free to revise it, if needed.

1) 문제해결

satable 이란 같은 key 값이 있을 때 input 순서와 동일한 순서로 정렬된 것을 의미한다. 선택정렬은 unstable 한 정렬인데 그 이유는 순서와 상관없이 swap 을 하기 때문이다 stable 한 정렬을 하기 위해서 swap 대신 삽입 정렬에서 사용하는 방법을 사용하면 stable 한 정렬을 할 수 있다.

먼저 data 값 중 가장 작은 값의 인덱스를 찾고 변수에 data 를 저장한다 그 후 가장 작은 값을 한 칸씩 왼쪽으로 이동시킨다. 변수에 저장한 data 를 i 번째 array 에 저장하고 다음으로 작은 값을 찾고 앞에 과정을 반복한다.

2) 실행결과



3 번

3. (Programming: **60 points**)
The radix sort was implemented using queue in p51 of the lecture note. Revise this code by revising the counting sort in p42 instead of the queue.

Step 1) Generate the input data to be sorted randomly.
# of input data: 1000 integers (data range: 0 ~2^{24}-1, i.e., 24-bit positive integer)
Step 2) Divide each data into 4 digits (=segments).
Step 3) Sort values at each digit from LSD to MSD.

Namely, in p47 of 'DS-Lec09-Sorting.pdf', n=1000, b=24, r=6, b/r=4 (# of digits)

Note) Consider what the data structure is needed to implement the radix sort using the counting sort.

```
RadixSort(A, d)
      for i=1 to d
         Counting Sort(A) on digit i
```

1) 문제해결

radix sort 를 counting sort 를 이용해 구현한다 1000 개의 input data 를 만들어야 하는데 rand() 함수는 2^15 까지 표현이 된다 2^24 까지 표현하기 위해 rand()를 두 번 사용해 2^24-1 까지 나오게 했다 24bit 를 4bits 로 나누어 6bit 6bit 6bit 6bit 으로 나누고 counting sort 로 구현을 시도하였다. Counting sort 함수에서 (list[j]>>(d*6))&(63)가 c 의 인덱스를 갖고 pdf, cdf 를 만들어서 정렬을 한다. 정렬된 b array 는 stable 하기 때문에 그 다음 6bit 을 기준으로 정렬 후 24bit 까지 정렬이 끝나면 오름차순의 정렬을 얻을 수 있다고 생각하였다. 하지만 정렬이 되지 않았고 여러 방법을 시도하였지만 구현이 되지 않았다.