**<과제>**

  주교재의 심층9-1(합병 정렬과 결정적 퀵 정렬 비교)

**<프로그램 작성과 실행예>**

  1) 입력 리스트는 **n** = **1,000,000**의 **배열**로 할 것.

  2) 저사양 PC 사용으로 인해 너무 느린 경우에 한해 **n** 값 적절히 축소 가능.

  3) 프로그램 구현 초기에 작은 **n**값을 사용하여 정렬이 정확히 수행되는지 확인한 후, 구현 완성 단계에서 큰 **n**값을 사용할 것.

  4) 입력 리스트 데이터를 얻기 위해 C 언어의 난수발생 함수를 활용하라.

  5) 입력 리스트로 “무작위한” 것과, “거의 정렬된” 것과, “거의 역정렬된” 것을 각각 1회씩 사용, 총 3가지의 상이한 입력에 대한 두 알고리즘의 실행예를 보여야 한다.

      - 따라서 총 6회의 정렬을 실행해야 한다.

      - “거의 정렬된“ 및 “거의 역정렬된“ 입력은 응용문제 8.5.3을 참고하여 마련하라.

      - 6회 각각의 정렬을 실행한 직후 해당 정렬의 수행시간을 한 라인씩 출력하라.

      - 이때, 수행시간 = 0 으로 나타나지 않도록, 수행시간의 소수점 이하 자릿수들을 충분히 명세해야 한다("수행시간 = 0" 또는 "수행시간 = 0.000000" 등이 포함된 출력을 제출하면 무조건 1점 감점).

      - 정렬 전후의 리스트 내용은 양이 많은 관계상 출력하지 않는다.

**<힌트>**

     global integer **n** = **1000000**

Alg **main**()

     1. (**A**, **B**, **C**, **A1**, **B1**, **C1**) ← **createArrays**()

     2. **inPlaceQuickSort**(**A**, **0**, **n** – **1**)       {deterministic version}

     3. **printCPUTime**()

     4. **mergeSort**(**A**1)                               {merge sort}

     5. **printCPUTime**()

     6. **inPlaceQuickSort**(**B**, **0**, **n** – **1**)       {deterministic version}

     7. **printCPUTime**()

     8. **mergeSort**(**B1**)                               {merge sort}

     9. **printCPUTime**()

     10. **inPlaceQuickSort**(**C**, **0**, **n** – **1**)     {deterministic version}

     11. **printCPUTime**()

     12. **mergeSort**(**C1**)                             {merge sort}

     13. **printCPUTime**()

     14. return

     Alg **createArrays**()

          input none

          output array **A**, **B**, **C**, **A1**, **B1**, **C1**of size **n**

     1. for **i**  ← **0** to **n** – **1**

**A**[**i**] ←  a random integer between **1** and **n**

     2. **B**, **C**  ←  **copyArray**(**A**)

     3. **inPlaceQuickSort**(**B**, 0, **n** – **1**)        {almost-sort version}

     4. **inPlaceQuickSort**(**C**, 0, **n** – **1**)        {almost-inverse-sort version}

     5. **A1**  ←  **copyArray**(**A**)

     6. **B1**  ←  **copyArray**(**B**)

     7. **C1**  ←  **copyArray**(**C**)

     8. return (**A**, **B**, **C**, **A1**, **B1**, **C1**)

†**주의:** **inPlaceQuickSort**가 호출하는 **inPlacePartition**은 입력리스트에 중복 키가 존재하더라도 분할을 수행하는 버전이어야 한다(심층8-2 참고).