ㅣ

jeon, sihyeon (전시현)

# 과제: HW-2

* Code Test: <https://www.acmicpc.net/problem/11004>
* Code Test: <https://www.acmicpc.net/problem/9251>
* 해당 프로그램에 주석 (comment)를 한글로 단다. (추가할 수 있는 것은 최대한 추가)

# #HW 2-1

|  |  |
| --- | --- |
|  | Quick sort를 응용하여 k번째 수를 출력한다. |

**Requirements**

* < 요구조건 #1: 간략한 핵심 요구 조건 설명>

제한 시간 내에 완료해야 한다.

## SOURCE CODE

|  |
| --- |
| /\*\*\*\*      \*file : 2quicksort.c      \*      \* Author : Jeon Sihyeon (wjstlgus13@gachon.ac.kr)      \*      \* Date : 09/20/2022      \*      \* Partner : I worked alone      \* Course : Algorithm (14465\_001)      \*      \* Summary of File:      \*   This file contains code which rearrange numbers and find k-th number.      \*  \*/  #include <stdio.h>  #include <stdlib.h>  void quickSort(int left, int right, int k);  int partition(int p, int r);  void swap(int \*a, int \*b);  int arr[5000000];  int main(){      int numofArr, k;      scanf("%d%d", &numofArr, &k);      for (int i = 0; i<numofArr; i++){          scanf("%d", &arr[i]);   //숫자 arr에 저장      }        quickSort(0, numofArr-1, k-1);  //quickSort function 호출 k번째이기 때문에 인덱스로 k-1 전달        printf("%d\n", arr[k-1]);       //k번째 있는 숫자 출력      return 0;  }  /\*\*   \* void quickSort(int left, int right, int k)   \*   \* quick sort를 응용하여 k번째 수를 알아내기 위해 최소한의 quicksort만 할 수 있도록 한다.   \*   \* parameters : int형   \*   \* return value : 없음   \*   \*/  void quickSort(int left, int right, int k){      int pivot = partition(left, right); //partition 함수 호출      if (pivot > k && left<pivot-1)      //k의 값에 따라 pivot을 기준으로 왼쪽, 또는 오른쪽 중 quickSort할 것을 선택          quickSort(left, pivot-1, k);    //quickSort 호출, left부터 pivot-1까지를 다시 quickSort. (recursive)      if(pivot<=k && pivot<right)          quickSort(pivot, right, k);     //quickSort 호출, pivot부터 right까지를 다시 quickSort. (recursive)  }  /\*\*   \* int partition(int left, int right)   \*   \* left와 right를 받아 pivot을 정한 뒤 좌우로 pivot보다 작은 수는 왼쪽,   \*  큰 수는 오른쪽으로 올 수 있도록 한다.   \*   \* parameters : int형   \*   \* return value : int형   \*   \*/  int partition(int left, int right){      int mid = (int)((left+right)/2);      int pivot = arr[mid];       //중간에 있는 숫자로 pivot을 설정      int index = left;      while(index<=right){          while(arr[index]<pivot){              index++;            //arr[i]가 pivot 이상일 때 까지 i++          }          while(pivot < arr[right]){              right--;        //arr[right]가 pivot 이하일 때까지 right--          }          if(index<=right){              swap(&arr[index], &arr[right]); //swap 함수를 call한다.              index++;              right--;          }      }      return index;   //index 값을 반환  }  /\*\*   \* void swap(int \*a, int \*b)   \*   \* 포인터를 이용하여 두 숫자의 값을 변경한다.   \*   \* parameters : int pointer형   \*   \* return value : 없음   \*   \*/  void swap(int \*a, int \*b){      int temp = \*a;      \*a = \*b;      \*b = temp;  } |

## OUTPUT (Screen Shots)

|  |  |
| --- | --- |
| #1 | [제출 결과] |
| (유의사항:제출 결과에 대한 분석 및 간단 설명) | |

홀수 개수의 숫자들 중 앞부분의 k번째를 구할 때

텍스트, 화면이(가) 표시된 사진

자동 생성된 설명

홀수 개수의 숫자들 중 뒷부분의 k번째를 구할 때

텍스트, 화면, 닫기, 이미지이(가) 표시된 사진

자동 생성된 설명

짝수 개수의 숫자들 중 앞부분의 k번째를 구할 때 (중복된 숫자도 포함)

텍스트, 화면, 이미지, 닫기이(가) 표시된 사진

자동 생성된 설명

짝수 개수의 숫자들 중 뒷부분의 k번째를 구할 때 (중복된 숫자도 포함)

텍스트, 화면, 이미지, 장치이(가) 표시된 사진

자동 생성된 설명

중복된 숫자가 여러 개일 때

텍스트, 밤하늘이(가) 표시된 사진

자동 생성된 설명

# #HW 2-2

|  |  |
| --- | --- |
|  | * Lcs의 값을 구한다. |

**Requirements**

* < 요구조건 #1: 간략한 핵심 요구 조건 설명>

시간 내에 완료해야 한다.

## SOURCE CODE

|  |
| --- |
| '''      lcs 함수:      LCS 알고리즘을 이용하여 longest common subsequence 값을      리턴해준다.  '''  def lcs(str1, str2):      strlen1 = len(str1)      strlen2 = len(str2)      c = [[None]\*(strlen2+1)for i in range(strlen1+1)]        for i in range(strlen1+1):  #i는 0부터 strlen1까지 for loop          for j in range(strlen2+1):  #j는 0부터 strlen2까지 for loop              if i==0 or j==0:                  c[i][j] = 0         #i와 j중 어느 인덱스라도 0이 되면 0을 저장한다.              elif (str1[i-1]==str2[j-1]):    #만약 같은 문자를 찾으면                  c[i][j] = c[i-1][j-1]+1     #c[i-1][j-1]의 값에 1을 더한다              else:                           #이를 제외하여                  c[i][j] = max(c[i-1][j], c[i][j-1]) #c[i-1][j], c[i][j-1] 중 더 큰 값을 c[i][j]에 저장한다.      return c[strlen1][strlen2]  '''      max 함수 :          더 크거나 같은 값을 return 해주는 함수이다.  '''  def max(a, b):      if (a>=b):          return a      return b  x = input()  y = input() #문장 2개 받기  print(lcs(x, y))    #lcs function을 call한 후 return 값을 출력 |

## OUTPUT (Screen Shots)

|  |  |
| --- | --- |
| #1 | [제출 결과] |
| (유의사항:제출 결과에 대한 분석 및 간단 설명) | |

//(본인이 실행한 화면을 출력)  
//(필요한 만큼 더 생성하여 실행 결과를 잘 파악할 수 있도록 제시)   
// (Screen shot이 없는 경우 생략 가능)

// 제출하는 문제 수에 맞게 위에 양식을 복사하여 사용

두 문장의 문자 개수가 1개씩인 경우

텍스트이(가) 표시된 사진

자동 생성된 설명

한 문장의 문자 개수가 1개인 경우

텍스트이(가) 표시된 사진

자동 생성된 설명

일반적인 경우1

텍스트이(가) 표시된 사진

자동 생성된 설명

일반적인 경우2

텍스트이(가) 표시된 사진

자동 생성된 설명

아무것도 매칭되지 않는 경우

텍스트이(가) 표시된 사진

자동 생성된 설명