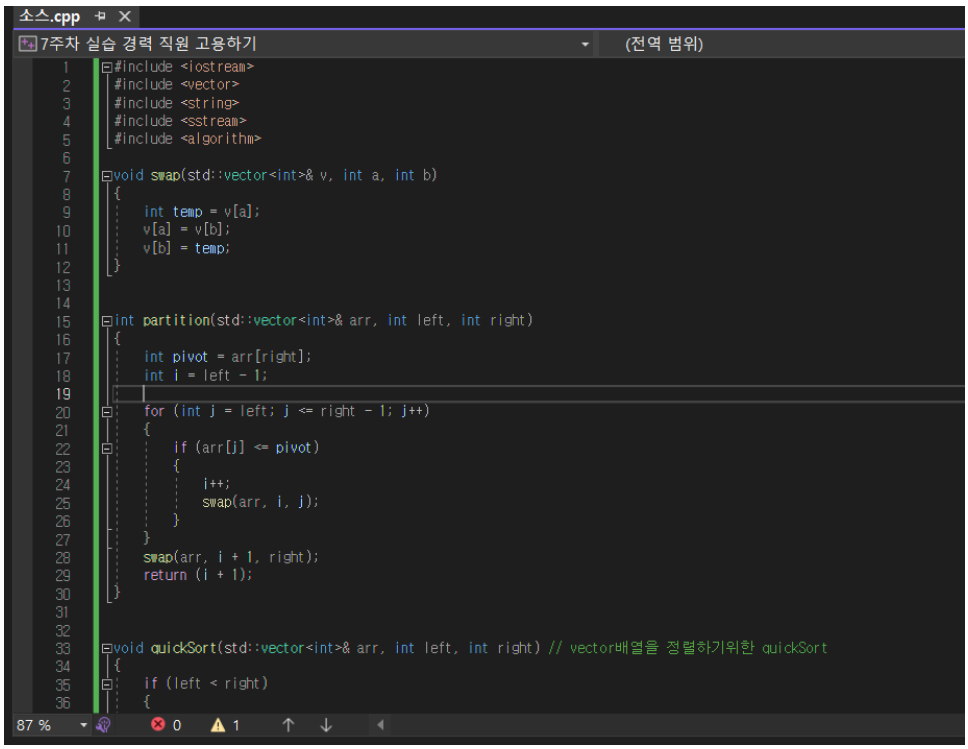
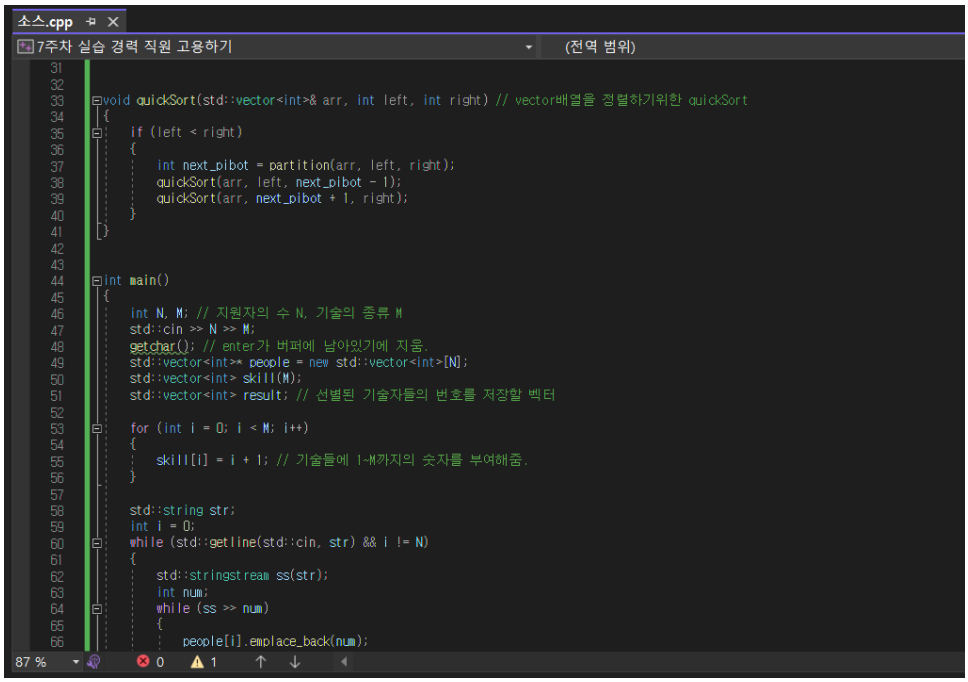
20101576 정재찬

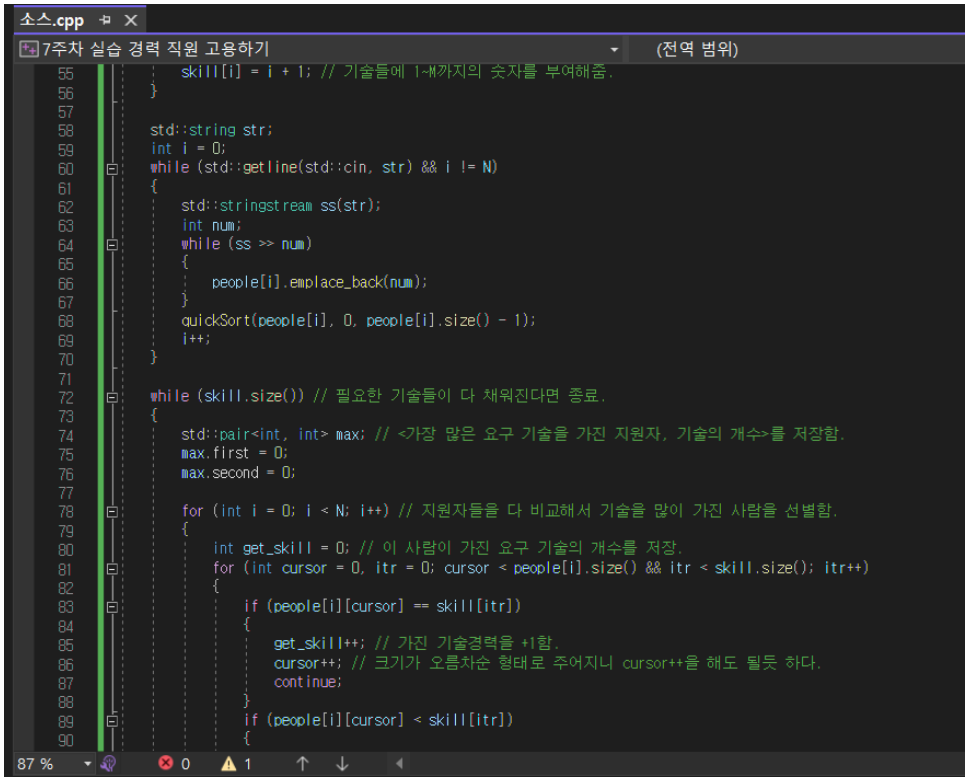
소스코드 및 주석(코드 설명)



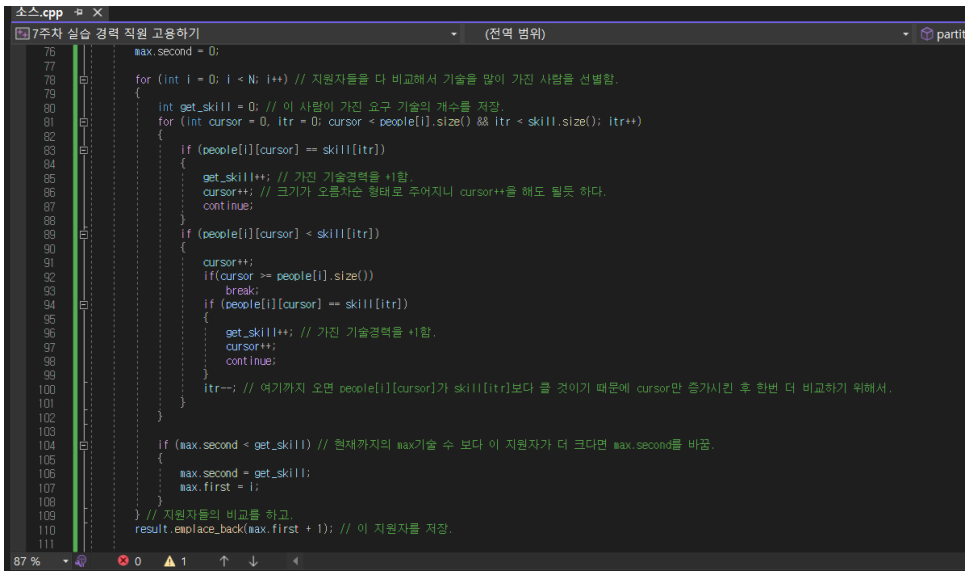
```cpp
#include <iostream>
#include <vector>
#include <string>
#include <sstream>
#include <algorithm>

void swap(std::vector<int>& v, int a, int b)
{
    int temp = v[a];
    v[a] = v[b];
    v[b] = temp;
}


int partition(std::vector<int>& arr, int left, int right)
{
    int pivot = arr[right];
    int i = left - 1;

    for (int j = left; j <= right - 1; j++)
    {
        if (arr[j] <= pivot)
        {
            i++;
            swap(arr, i, j);
        }
    }
    swap(arr, i + 1, right);
    return (i + 1);
}


void quickSort(std::vector<int>& arr, int left, int right) // vector배열을 정렬하기위한 quickSort
{
    if (left < right)
    {
```



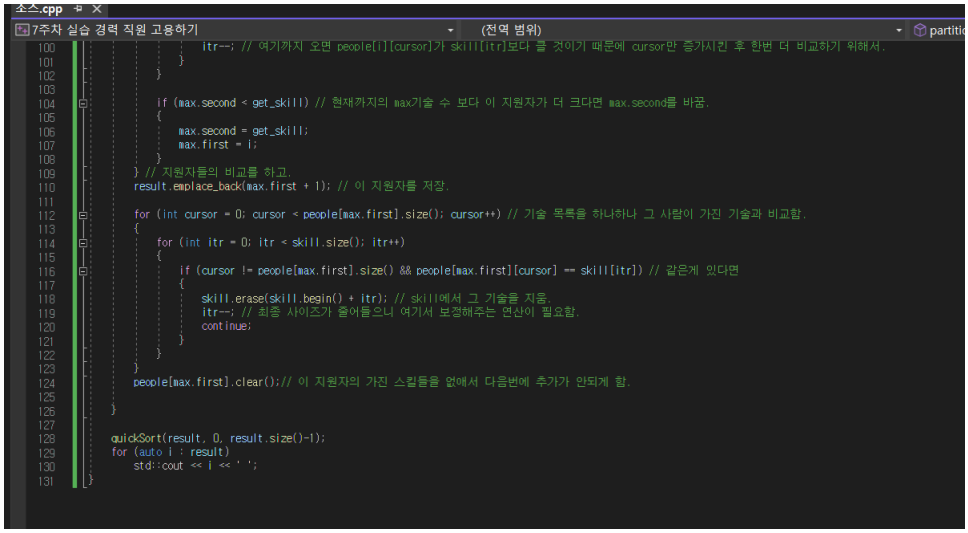
```cpp


void quickSort(std::vector<int>& arr, int left, int right) // vector배열을 정렬하기위한 quickSort
{
    if (left < right)
    {
        int next_pibot = partition(arr, left, right);
        quickSort(arr, left, next_pibot - 1);
        quickSort(arr, next_pibot + 1, right);
    }
}


int main()
{
    int N, M; // 지원자의 수 N, 기술의 종류 M
    std::cin >> N >> M;
    getchar(); // enter가 버퍼에 남아있기에 지움.
    std::vector<int>* people = new std::vector<int>[N];
    std::vector<int> skill(M);
    std::vector<int> result; // 선별된 기술자들의 번호를 저장할 벡터

    for (int i = 0; i < M; i++)
    {
        skill[i] = i + 1; // 기술들에 1~M까지의 숫자를 부여해줌.
    }

    std::string str;
    int i = 0;
    while (std::getline(std::cin, str) && i != N)
    {
        std::stringstream ss(str);
        int num;
        while (ss >> num)
        {
            people[i].emplace_back(num);
```




```
55          skill[i] = i + 1; // 기술들에 1~M까지의 숫자를 부여해줌.
56      }
57
58      std::string str;
59      int i = 0;
60      while (std::getline(std::cin, str) && i != N)
61      {
62          std::stringstream ss(str);
63          int num;
64          while (ss >> num)
65          {
66              people[i].emplace_back(num);
67          }
68          quickSort(people[i], 0, people[i].size() - 1);
69          i++;
70      }
71
72      while (skill.size()) // 필요한 기술들이 다 채워진다면 종료.
73      {
74          std::pair<int, int> max; // <가장 많은 요구 기술을 가진 지원자, 기술의 개수>를 저장함.
75          max.first = 0;
76          max.second = 0;
77
78          for (int i = 0; i < N; i++) // 지원자들을 다 비교해서 기술을 많이 가진 사람을 선별함.
79          {
80              int get_skill = 0; // 이 사람이 가진 요구 기술의 개수를 저장.
81              for (int cursor = 0, itr = 0; cursor < people[i].size() && itr < skill.size(); itr++)
82              {
83                  if (people[i][cursor] == skill[itr])
84                  {
85                      get_skill++; // 가진 기술경력을 +1함.
86                      cursor++; // 크기가 오름차순 형태로 주어지니 cursor++을 해도 될듯 하다.
87                      continue;
88                  }
89                  if (people[i][cursor] < skill[itr])
90                  {
```




```
76          max.second = 0;
77
78          for (int i = 0; i < N; i++) // 지원자들을 다 비교해서 기술을 많이 가진 사람을 선별함.
79          {
80              int get_skill = 0; // 이 사람이 가진 요구 기술의 개수를 저장.
81              for (int cursor = 0, itr = 0; cursor < people[i].size() && itr < skill.size(); itr++)
82              {
83                  if (people[i][cursor] == skill[itr])
84                  {
85                      get_skill++; // 가진 기술경력을 +1함.
86                      cursor++; // 크기가 오름차순 형태로 주어지니 cursor++을 해도 될듯 하다.
87                      continue;
88                  }
89                  if (people[i][cursor] < skill[itr])
90                  {
91                      cursor++;
92                      if(cursor >= people[i].size())
93                          break;
94                      if (people[i][cursor] == skill[itr])
95                      {
96                          get_skill++; // 가진 기술경력을 +1함.
97                          cursor++;
98                          continue;
99                      }
100                     itr--; // 여기까지 오면 people[i][cursor]가 skill[itr]보다 클 것이기 때문에 cursor만 증가시킨 후 한번 더 비교하기 위해서.
101                 }
102             }
103
104         if (max.second < get_skill) // 현재까지의 max기술 수 보다 이 지원자가 더 크다면 max.second를 바꿈.
105         {
106             max.second = get_skill;
107             max.first = i;
108         }
109     } // 지원자들의 비교를 하고.
110     result.emplace_back(max.first + 1); // 이 지원자를 저장.
111
```



#시간 복잡도 계산

1. while(skill.size()) : 전체집합의 원소가 0이될때까지 : 최대 O(M); // M개의 기술
2. For(N번) : N명의 기술자들을 순회하며 비교하는 반복문 : O(N) : // N명
3. 내부 for문 : i번째 기술자의 기술들과 전체집합의 기술들을 비교함 : 최대 O(M+N)
4. 그래서 총 시간 복잡도는 : O(N^3) // 3차.