<고급C 프로그래밍 및 실습> 2차 인증시험 문제지

2020.11.19 (목)

※ 문제지의 무단 배포 및 사용을 원칙적으로 금지합니다.

- 특히, 커뮤니티, 개인 블로그 등 인터넷 사이트 게시를 절대 금지합니다.

※ 문제에 대한 안내

- 문제지는 총 13페이지이고, 총 5문제 500점 만점이고, 문제의 순서는 난이도와 관계없다.
- 특별한 언급이 없으면 문제의 조건에 맞지 않는 입력은 입력되지 않는다고 가정하라.
- 입출력 예시에서 ↦ 이 후는 각 입력과 출력에 대한 설명이다.
- OJ에서 Sample Submit 기능사용가능하며, 점수에는 포함되지 않는다.

※ 주의: [문제 1-1]과 [문제 1-2]는 연관된 문제이다. [문제 1-1] 만 풀면 50점, [문제 1-2] 만 풀어도 100점이다. 가장 높은 배점 하나만 반영한다.

[문제 1-1] (50점) N명 학생의 이름과 1차, 2차 고급C 인증시험 점수를 입력 받는다. 1차 인증 기준 점수 N1 이하 **혹은** 2차 인증 기준 점수 N2 이하를 받은 학생을 모두 화면에 출력하시오. 단, N≤100이다.

- insertData 함수
  - 함수원형 : **void insertData(struct student_info *s, int N)**
  - N명 학생의 이름과 학생 당 두 번의 인증 시험 점수를 입력 받고, 학생의 인증시험 평균 점수를 계산하여 **구조체 배열**에 저장한다.
  - 학생의 이름은 공백을 포함하지 않는 최대 10개 영어 문자이다.
- printResult 함수
  - 함수원형 : **void printResult(struct student_info *s, int N, int N1, int N2)**
  - 각각의 인증시험에서 기준 이하의 점수를 받은 학생들의 이름과 인증 점수를 출력 한다. **출력 순서는 입력된 순서**이다.
- insertData와 printResult 함수에서는 구조체 배열 참조 시 배열 표기 [ ]의 사용을 금지하고, 배열을 반복문으로 훑어볼 시, 주소를 이용하여 포인터를 이동시키며, 반복문을 구현 한다.

▶ 함수를 사용하지 않거나 함수 원형을 수정한 경우 **(100% 감점)**

▶ main 함수에서 출력을 한 경우 **(30% 감점)**

▶ insertData, printResult 함수에서 배열 표기 **[ ]**를 사용한 경우 **(각 15% 감점)**

▶ insertData, printResult 함수에서 배열을 반복문으로 훑어볼 시, 주소를 이용하여 포인터를 이동시키며, 반복문 구현 **(위반 시 각 15% 감점)**

▶ 전역변수를 사용한 경우 **(100% 감점)**

입력 예시 1	출력 예시 1
6 60 50 ↦ N, 1차 기준 점수 N1, 2차 기준 점수 N2 Kim 80 46 Choi 76 98 Park 64 89 Chung 87 100 Lee 40 28 Kang 65 77	Kim 80 46 Lee 40 28

[문제 1-2] (100점) N명 학생의 이름과 1차, 2차 고급C 인증시험 점수를 입력 받아 인증시험 전체 평균점수와 **평균점수 이하인** 학생의 수를 출력 한 후, 평균 점수 이하인 학생의 이름과 점수를 입력된 순서로 출력하는 프로그램을 아래의 함수를 사용하여 작성하시오. 단, N≤100이고, **각 학생의 인증시험 평균 점수는 1차 인증시험 점수의 40%와 2차 인증시험 점수의 60%의 합으로 계산**한다.

- insertData 함수
  - 함수원형 : **void insertData(struct student_info *s, int N)**
  - N명 학생의 이름과 학생 당 두 번의 인증 시험 점수를 입력 받고, 학생의 인증시험 평균 점수를 계산하여 구조체 배열에 저장한다.
  - 학생의 이름은 공백을 포함하지 않는 최대 10개 영어 문자이다.
- getAllavg 함수
  - 함수원형 : **double getAllavg(struct student_info s[], int N)**
  - 전체 학생의 **인증시험 평균 점수와 평균 이하의 점수를 받은 학생의 수를** 계산하여 출력하고, 인증시험 전체 평균 점수를 반환한다. 평균 점수는 소수점 이하 둘째자리까지 출력한다.
- printResult 함수
  - 함수원형 : **void printResult(struct student_info *s, int N, double avg)**
  - 인증시험 전체 평균 이하의 점수를 받은 학생들의 이름과 학생 각자의 평균 점수를 출력한다. **출력 순서는 입력된 순서**이며, 평균 점수는 소수점 이하 둘째자리까지 출력한다.
- insertData와 printResult 함수에서는 구조체 배열 참조 시 배열 표기 [ ]의 사용을 금지하고, 배열을 반복문으로 훑어볼 시, 주소를 이용하여 포인터를 이동시키며, 반복문을 구현 한다.

▶ 함수를 사용하지 않거나 함수 원형을 수정한 경우 **(100% 감점)**
▶ main 함수에서 출력을 한 경우 **(30% 감점)**
▶ main 함수에서 전체 평균을 계산한 경우 **(30% 감점)**
▶ insertData, printResult 함수에서 배열 표기 **[ ]**를 사용한 경우 **(각 15% 감점)**
▶ insertData, printResult 함수에서 배열을 반복문으로 훑어볼 시, 주소를 이용하여 포인터를 이동시키며, 반복문 구현 **(위반 시 각 15% 감점)**
▶ 전역변수를 사용한 경우 **(100% 감점)**

입력 예시 1	출력 예시 1
6 Kim 80 46 Choi 76 98 Park 64 89 Chung 87 100 Lee 40 28 Kang 65 77	71.27 2 ↦ 인증시험 평균점수와 **평균 이하**인 학생 수 Kim 59.60 ↦ 전체 평균 이하의 점수를 받은 학생들의 Lee 32.80 이름과 학생 각자의 평균 점수 ↦ 1차 인증시험 40% + 2차 인증시험 60%인 경우, 각 학생의 평균 점수는 순서대로, 59.60, 89.20, 79.00, 94.80, 32.80, 72.20 이다.

※ 주의: [문제 2-1]과 [문제 2-2]는 연관된 문제이다. [문제 2-1] 만 풀면 50점, [문제 2-2] 만 풀어도 100점이다. 가장 높은 배점 하나만 반영한다.

[문제 2-1] (50점) 공백이 포함 된 N개의 문자열을 입력받아, 각 문자열의 **M1번째 문자부터 M2번째 문자**까지의 문자들 중에 **영어대문자가 몇 개** 포함되는지 세어 (M1번째 문자와 M2번째 문자 포함), 영어 대문자의 개수 출력하는 프로그램을 아래의 check_u 함수를 사용하여 작성하시오. 단, **문자열의 최대 길이는 100**이고, **영어대문자를 하나도 포함하지 않은 문자열의 대문자 개수는 출력에서 제외**한다. (N은 최대 20 이다. 동적할당문제가 아님. 2차원 배열이 필요 없는 문제이다)

- 정수 N과 M1, M2를 입력받고, 공백이 포함된 N개의 문자열을 입력 받는다.
- M1과 M2는 0부터 표시되어 문자열내의 각 문자의 위치를 나타낸다. M1 값이 문자열의 길이보다 큰 경우는 없다고 가정한다. **M2값이 문자열의 길이보다 큰 경우에는 문자열 끝까지만 검사한다**. M1 ≦ M2
- 필요한 경우, 표준 문자열 함수를 사용한다.

- check_u 함수
  - 함수원형 : **int check_u(char *p, char *q)**
  - 문자열 중 p가 가리키는 문자부터 q가 가리키는 문자까지 영어대문자의 수를 세어 반환한다. **(p가 가리키는 문자와 q가 가리키는 문자 포함하여 검사)**
  - 함수에서 **반복문으로 배열을 훑어볼 시, 주소를 이용하여 반복문을 구현한다.** 즉, 포인터가 배열의 각 원소를 순차적으로 가리키도록 하며, 포인터가 가리키는 위치의 원소에 대해 필요한 작업을 수행한다. p[i] 또는 p+i 표기 사용 금지
- 단, **main 함수에서의 주소 표현 방식은 제한 사항 없다.**

▶ check_u 함수를 사용하지 않거나 함수 원형을 수정한 경우 **(100% 감점)**

▶ check_u 함수에서 포인터 주소 표기를 잘 못 사용한 경우 **(p[i] 또는 p+i 표기 사용 금지) (50% 감점)**

▶ 전역변수를 사용한 경우 **(100% 감점)**

입력 예시 1	출력 예시 1
5 ↦ N = 문자열 줄 수 3 8 ↦ M1번째 문자부터 M2까지 대문자 ADVANCED C PROGRAMMING PROGRAM Good ↦ 대문자 개수가 0이면 출력에서 제외 ****C LANGUAGE THIS IS STRING ↦ 해당 구간 문자는 진하게 표시	5 4 4 4

[문제 2-2] (100점) 공백이 포함 된 N개의 문자열을 입력받아, 각 문자열의 M1번째 문자부터 M2번째 문자까지의 문자들 중에 영어대문자가 몇 개 포함되는지 세어 (M1번째 문자와 M2번째 문자 포함), **영어 대문자를 가장 적게 포함하는 문자열을 출력**하는 프로그램을 아래의 check_u 함수를 사용하여 작성하시오. 단, **문자열의 최대 길이는 100**이고, **영어대문자를 하나도 포함하지 않은 문자열은 출력에서 제외**한다. (N은 최대 20 이다. 동적할당문제가 아님. 2차원 배열이 필요 없는 문제이다)

- 정수 N과 M1, M2를 입력받고, 공백이 포함된 N개의 문자열을 입력 받는다.
- M1과 M2는 0부터 표시되어 문자열내의 각 문자의 위치를 나타낸다. M1 값이 문자열의 길이보다 큰 경우는 없다고 가정한다. **M2값이 문자열의 길이보다 큰 경우에는 문자열 끝까지만 검사한다**. M1 ≦ M2
- 영어대문자를 가장 적게 포함하는 문자열이 두 개 이상인 경우에는 **해당 문자열 중 문자열의 길이가 가장 긴 문자열을 출력**한다.
- 영어대문자를 가장 적게 포함하는 문자열이 두 개 이상이며, 이 중 길이가 같은 문자열이 있는 경우, **해당 문자열 중 가장 나중에 입력된 문자열을 출력**한다.
- 필요한 경우, 표준 문자열 함수를 사용한다.
- N개의 문자열 중에서, 대문자를 1개 이상 포함하는 문자열이 1개 이상 있다.

- check_u 함수
  - 함수원형 : **int check_u(char *p, char *q)**
  - 문자열 중 p가 가리키는 문자부터 q가 가리키는 문자까지 영어대문자의 수를 세어 반환한다. **(p가 가리키는 문자와 q가 가리키는 문자 포함하여 검사)**
  - 함수에서 **반복문으로 배열을 훑어볼 시, 주소를 이용하여 반복문을 구현한다.** 즉, 포인터가 배열의 각 원소를 순차적으로 가리키도록 하며, 포인터가 가리키는 위치의 원소에 대해 필요한 작업을 수행한다.
- 단, **main 함수에서의 주소 표현 방식은 제한 사항 없다.**

▶ check_u 함수를 사용하지 않거나 함수 원형을 수정한 경우 **(100% 감점)**

▶ check_u 함수에서 포인터 주소 표기를 잘 못 사용한 경우 **(p[i] 또는 p+i 표기 사용 금지) (50% 감점)**

▶ 전역변수를 사용한 경우 **(100% 감점)**

입력 예시 1	출력 예시 1
5 3 8 ADV**ANCED** C PROGRAMMING PRO**GRAM** Good ****C LAN**GUAGE THI**S IS S**TRING ↦ 해당 구간 문자는 진하게 표시	THIS IS STRING ↦ 대문자 개수가 0인 Good을 제외하면 최소 대문자 개수는 4이다. ↦ 이 중 문자열 전체 길이가 가장 긴 ****C LANGUAGE와 THIS IS STRING 중 나중에 입력된 THIS IS STRING 출력

※ [문제 3-1]에서 [문제 3-2]까지 연관된 문제이며, 배점이 각각 50, 100점이다. [문제 3-1]을 안 풀고 [문제 3-2]만 풀어도 100점을 부여한다. 가장 높은 배점 하나만 반영한다. 합산하지 않는다.

[문제 3-1] (50점) 동적할당 문제가 아니고 문자열 문제이다.

(1) 첫 번째 줄에서, N개의 단어를 포함하는 문자열 A를 입력받고

(2) 두 번째 줄에서, M개의 단어를 포함하는 문자열 B를 입력받아

(3) 두 문자열 모두에 포함된 단어의 수를 화면에 출력하고,

(4) 두 문자열 모두에 포함된 단어 중 가장 길이가 긴 단어를 화면에 출력하시오.
만약, 두 문자열 모두에 포함된 단어 중 가장 길이가 긴 단어가 두 개 이상 존재하면, 사전 순서 상 앞선 단어를 출력한다.

(요약) **두 문자열 모두**에 포함된 단어 중 --- 가장 **길이**가 긴 단어 --- **사전** 순서 상 앞선 단어

- 문자열 A와 B의 최대 길이는 100이고, N과 M은 20보다 크지 않다.
- 단어는 알파벳 소문자로만 구성되며, 문자숫자나 특수문자를 포함하지 않는다.
- 단어와 단어 사이에는 공백 문자가 하나만 있으며, 문자열의 첫 문자가 공백 문자인 경우는 없다.
- 문자열 A에는 동일한 단어가 반복해서 나타나지 않으며, 문자열 B에도 동일한 단어가 반복해서 나타나지 않는다.
- 문자열 A와 문자열 B에 모두 포함되는 단어가, 최소 1개 이상이다.

▶ 강의노트에 있는 문자열 처리 함수 중 2가지 이상을 의미 있는 작업에 반드시 사용 **(위반 시 각 25% 감점)**

▶ strtok 와 같이 수업 시간에 다루지 않은 표준 문자열 처리 함수 사용금지 **(위반 시 50% 감점)**

입력 예시 1	출력 예시 1
red orange yellow green blue purple pink red yellow black white purple	3 purple
↦ 두 문자열에 공통으로 나타나는 단어는 3개이고, 3 단어 중 가장 긴 단어는 yellow와 purple이며, 이 중 사전 순서상 앞선 단어인 purple 출력	

[문제 3-2] (100점) 동적할당 문제가 아니고 문자열 문제이다.

(1) 첫 번째 줄에서, N개의 단어를 포함하는 문자열 A를 입력받고

(2) 두 번째 줄에서, M개의 단어를 포함하는 문자열 B를 입력받아

(3) 문자열 A에 포함된 **두 개의 단어를 합성**하여 만들어진 단어 중, 문자열 B에 포함되어 있는 단어의 수를 출력하고

(4) 문자열 B에 포함되어 있는 합성된 단어 중 가장 길이가 긴 단어를 출력하시오.

만약, 문자열 B에 포함된 합성된 단어 중 가장 길이가 긴 단어가 두 개 이상 존재하면, 사전 순서 상 앞선 단어를 출력한다.

예를 들어, 문자열 A에 "orange"와 "pink"라는 단어가 있고, 문자열 B에 "orangepink"라는 단어가 있으면, 문자열 B에 포함된 합성된 단어가 된다. 또한, 문자열 B에 "pinkorange"가 있어도 문자열 B에 포함된 합성된 단어가 된다. 단, "orangeorange"와 같이 한 단어를 반복해서 합성하지는 않는다.

- 문자열 A와 B의 최대 길이는 100이고, N과 M은 20보다 크지 않다.
- 단어는 알파벳 소문자로만 구성되며, 문자숫자나 특수문자를 포함하지 않는다.
- 단어와 단어 사이에는 공백 문자가 하나만 있으며, 문자열의 첫 문자가 공백 문자인 경우는 없다.
- **문자열 A에는 동일한 단어가 반복해서 나타날 수 있으며, 문자열 B에도 동일한 단어가 반복해서 나타날 수 있다.**
- **문자열 A에 "orange"라는 단어가 두 개 있어도, "orangeorange"로 합성하지 않는다. 만약, 문자열 A에 "orange"와 "pink"라는 단어가 있고, 문자열 B에 "orangepink"라는 단어가 두 개 있어도, 문자열 B에 포함된 합성된 단어는 하나이다.**
- **문자열 A에 "or"와 "ange"의 단어가 있어 합성어 "orange"가 만들어 지고, 또한 "ora"와 "nge"의 단어가 있어, 동일한 합성어 "orange"가 만들어지는 경우, 문자열 B에 나타나는 합성어는 하나로 센다.** (출력 예시 3 참고)
- 문자열 A에 포함된 단어로 합성한 단어 중, 문자열 B에 포함되는 단어는 최소 1개 이상이다.

▶ 강의노트에 있는 문자열 처리 함수들 중 3가지 이상을 의미 있는 작업에 반드시 사용 **(위반 시 각 25% 감점)**

▶ strtok 와 같이 수업 시간에 다루지 않은 표준 문자열 함수 사용금지 **(위반 시 50% 감점)**

입력 예시 1

red orange yellow green blue purple pink red redorange orangered yellow black bluegreen white purple

출력 예시 1

3 bluegreen
↦ 문자열 A의 단어로 합성된 문자열 B의 단어는 3개이고, 3 단어의 길이가 동일하여, 사전 순서상 앞선 단어인 bluegreen 출력

입력 예시 2

red orange yellow **red** green blue purple pink red **redred** yellow black bluegreen white purple **bluegreen** bluepurple

출력 예시 2

2 bluepurple
↦ 문자열 A의 단어로 합성된 문자열 B의 단어는 2개이고, 2 단어 중 길이가 긴 bluepurple 출력 ↦ 문자열 A에 red가 2개 있지만, 동일 단어의 합성으로 만들어진 redred는 합성어로 판정되지 않음 ↦ 문자열 B에 bluegreen이 2개 있지만, blue와 green의 합성어는 하나로 판정되며, 중복 인정되지 않음

입력 예시 3

apple or ange pineapple coconuts lemons ora nge appleapple **orange orange** orlemons

출력 예시 3

2 orlemons
↦ 문자열 A의 or+ange와 ora+nge의 합성어 orange는 문자열 B에 두 개가 있어도 하나로 판정되며, 중복 인정되지 않음

※주의: [문제 4-1]과 [문제 4-2]는 연관된 문제이다. [문제 4-1] 만 풀면 50점, [문제 4-2] 만 풀어도 100점이다. 가장 높은 배점 하나만 반영한다.

[문제 4-1] (50점) 양의 정수 M을 입력받고, M의 약수가 20개 이하이면, 약수들을 가장 큰 수부터 내림차순으로 출력하고, M의 약수가 20개가 넘는 경우 "none"을 화면에 출력하는 프로그램을 동적할당과 num_of_divisors 함수를 이용하여 작성 하시오.

- main함수에서는 정수 M과 정수 포인터 arr을 선언
  - M을 입력받고, **크기 20의 정수 배열을 동적으로 할당받아, arr 포인터에 연결**
  - num_of_divisors 함수를 호출하고, M의 약수가 20개 이하인 경우, arr가 가리키는 동적 배

열에 저장된 값을 **내림차순으로 출력**

- M의 약수가 20개를 초과하는 경우에는 약수를 화면에 출력하지 않고 "none"을 출력

◦ 함수 num_of_divisors를 호출하여 약수들을 구한다.

- 함수원형 : **int num_of_divisors(int M, int *arr);**
- 반환값: 정수 M의 약수의 개수 또는 -1
- 하는 일 : arr에 M의 약수를 **작은 수부터 큰 수까지 차례로 저장하고 약수의 개수를 반환**한다. 단, 약수의 개수가 20개를 넘어가면 -1을 반환한다.

num_of_divisors 함수에서는 약수를 오름차순으로 저장하고, main 함수에서는 저장된 약수들을 거꾸로 출력하여, 내림차순으로 출력이 되도록 해야 한다.

▶ 동적할당 미사용 시 (100% 감점)
▶ 함수 내부에서 화면에 출력을 하거나, 전역변수 사용 시, 함수 미사용 시 (100% 감점)
▶ 함수의 반환값이나 함수인자의 정의와 다르게 사용하거나 문제의 기술과 다르게 한 경우 (30% 감점)
▶ 동적할당 후 오류 체크 및 해제 안하면 (각 20% 감점)

입력 예시 1	출력 예시 1
10	10 5 2 1□

입력 예시 2	출력 예시 2
1000 ↦ N	none

[문제 4-2] (100점) 양의 정수 N을 입력받고, N개의 정수를 입력받아, 각 정수의 약수를 동적으로 할당받은 포인터 배열에 연결된 정수 배열에 저장한다.

(1) 양의 정수 K를 입력받고, K번째 정수의 약수들을 큰 수부터 작은 수까지 내림차순으로 출력한다.

(2) 약수의 합이 제일 큰 정수를 출력한다.

아래의 설명을 잘 읽고 **조건에 맞게 구현**한다.

◦ main함수에서 변수 선언

int **darr, *num, temp[20];

이 외의 정적 또는 동적 배열은 사용할 수 **없음**. 배열 아닌 다른 변수는 사용 가능.

(1) 정수 N을 입력받고, 크기 N의 정수 포인터 배열을 동적으로 할당받아, darr에 연결한다.

별도로, 크기 N의 정수 배열을 동적으로 할당 받아, num에 연결한다.

(2) darr[i]에는 i번째 정수의 약수들이 저장된 동적으로 할당 받은 정수배열이 연결되고, num[i]에는 i번째 정수의 약수의 개수를 저장한다.

(3) (2)번 작업을 위해 num_of_divisors 함수를 사용한다.

- num_of_divisors 함수는 temp에 정수 M의 약수를 저장한 후, 약수의 개수를 반환하고, 이 반환 값은 num[i]에 저장된다.
- main 함수에서는 num[i] 크기의 정수배열을 동적으로 할당 받아, darr[i]에 연결하고, temp에 저장된 약수를 darr[i]에 연결된 배열에 차례로 옮긴다.
- 단, num_of_divisors에서 -1이 반환된 경우에는 이 작업을 하지 않고, num[i]에 –1을 저장한다.

(4) 양의 정수 K를 입력받아 K번째 정수의 약수들을 크기가 큰 약수부터 내림차순으로 화면에 출력한다. (**배열에 저장된 순서의 반대로 출력**) 단, 약수의 개수가 20개가 넘는 경우에는 –1을 화면에 출력 한다. (입력예시 3) (K는 0부터 시작 한다.)

(5) cal_sum 함수를 반복 호출하여 각 정수에 대해 저장된 약수의 합을 차례로 구하고, 약수의 합이 최대인 정수를 출력한다. 약수의 개수가 20개가 넘는 정수는 약수의 합을 구할 때 제외한다. N개 정수 모두 약수가 20개 넘는 경우는 없다고 가정한다.

(6) 메모리를 해제한다.

◦ 함수 num_of_divisors

- 함수원형 : int num_of_divisors(int M, int *arr);
- 반환값: 정수 M의 약수의 개수 또는 -1
- 하는 일 : arr에 M의 약수를 작은 수부터 큰 수 까지 차례로 저장하고, 약수의 개수를 반환한다. 단, 약수 개수가 20개를 넘어가면 –1을 반환한다.

◦ 함수 cal_sum

- 함수원형 : int cal_sum(int *onearr, int size);

반환값 : onearr의 배열 원소를 size개 더한 합.

- 하는 일 : 정수 배열의 시작 주소와 배열의 원소 개수를 인자로 받아, 배열 원소를 개수만큼 더한 합을 구하여 반환.

주의 : 1차원 정수 배열의 원소 합을 구하는 것으로 구현해야한다.

▶ num, darr 동적할당 미사용시 (각 20%감점)
▶ darr의 각 원소 동적할당 시 정확한 크기 할당을 하지 못한 경우(30% 감점)
▶ num, darr 크기 오류 (각 20% 감점)
▶ darr에 관련된 모든 동적 할당에 대해서 오류체크를 제대로 하지 않으면 (20% 감점)
(시간관계상 다른 동적할당에 대한 오류체크는 생략하여도 된다.)
▶ main 함수에서 추가 배열을 사용하면 (30% 감점)
▶ 함수의 반환값이나 함수인자 정의와 다르게 사용하면 (각 30%감점)
▶ 메모리 해제 제대로 안 한 경우 (num - 각 10%, darr - 30%감점)

▶ 그 이외 문제의 기술과 다르게 구현한 경우 - 불필요한 작업, 다른 처리방식등 (각 20% 감점)

입력 예시 1

```
3                ↦ N
10 20 30
1                ↦ K
```

출력 예시 1

```
20 10 5 4 2 1              ↦ 1번째 수 20의 약수
72                 ↦   1 2 3 5 6 10 15 30  (30의 약수)
                   ↦   약수의 합중에서 최대값이 72
```

입력 예시 2

```
5                ↦ N
100 500 1000 5 3
0                ↦ K
```

출력 예시 2

```
100 50 25 20 10 5 4 2 1     ↦ 0번째 수 100의 약수
2340          ↦  1000의 약수 합
```

입력 예시 3

```
3                ↦ N
1000 10000 100
1                ↦ K
```

출력 예시 3

```
-1          ↦ 1번째 수 10000의 약수개수가 20개 초과
2340
```

※주의: [문제 5-1]에서 [문제 5-2]까지 연관된 문제이다. [문제 5-1] 만 풀면 50점, [문제 5-2] 만 풀어도 150점이다. 가장 높은 배점 하나만 반영한다.

[문제 5-1] (50점) 인테리어 공사를 위해 필요한 작업들 중에 시작시간이 가장 빠른 작업을 찾는 프로그램을 작성하려고 한다. 양의 정수 N과, N개 작업의 작업 명, 시작 시간, 종료 시간, 작업 대금(수익)을 차례로 입력받아 구조체 배열에 저장하고, 이들 중에 가장 **시작시간이 빠른** 작업을 찾아 출력하는 프로그램을 작성한다. 단, 아래와 같은 구조체 자료형과 구조체 배열을 위한 동적 할당을 한다. 시작 시간이 동일한 경우는 입력 순서가 제일 빠른 작업을 출력한다.

1) 구조체 자료형

```
struct job{
   char *name;      // 작업명 (공백 없이 최대 100자)
   char *start;     // 작업 시작 시간 (HH:MM:SS; 24시간제 시간,분,초 예)15:32:35)
   char *end;       // 작업  종료 시간 (위와 동일)
   int money;       // 작업 대금 (수익)
} ;
```

- typedef를 사용하여 job 구조체의 type명을 ABC로 바꾸어 사용하시오.
- 구조체 멤버를 추가하지 마시오.

2) 구조체 내부의 name은 공백 없이 최대 100자인 문자로, 메모리를 절약하기 위해 문자배열을 동적 할당하여 사용하고 start, end는 24시간제 표기를 담을 수 있는 만큼 문자배열 동적 할당하여 사용한다. (예: 15:32:35 – 오후 3시 32분 35초) 예: 00:00:01

멤버 변수 name, start, end에 메모리 동적할당하여 연결할 때 정확한 크기로 할당하시오.

입력 예시 1

```
4     ↦ 작업 개수
Remove 09:45:00 13:00:00 50
Cleaning 08:27:45 10:00:00 50
Painting 13:00:15 18:00:00 100
Checking 15:00:00 18:00:00 10
```

출력 예시 1

```
Cleaning 08:27:45 10:00:00 50
```

입력 예시 2

```
5     ↦ 작업 개수
Remove 09:45:00 13:00:00 50
Cleaning 08:27:45 10:00:00 80
Painting 13:00:15 18:00:00 100
Checking 15:00:00 18:00:00 10
Preview 08:27:45 10:00:00 80
```

출력 예시 2

```
Cleaning 08:27:45 10:00:00 80
```

- 구조체 배열 동적할당 사용하지 않은 경우 (100% 감점)
- 메모리 동적 할당 오류체크 최소 1번 해야 함. 한 번도 안한 경우 (20% 감점)
- 동적 할당 받은 메모리 모두 해제 (30% 감점)
- 구조체 멤버 변수들 동적 할당 안하면 (name 20%감점, start/end 각각 10%감점)
- 구조체 멤버 변수들 동적 할당 할 때 크기 틀리면 (name/start/end 각각 5%감점)
- 구조체 멤버 추가하면 10% 감점
- typedef 사용하지 않으면 10% 감점

[문제 5-2] (100점) 인테리어 공사 중 하루에 작업을 3개만 진행하기로 결정하였다. 양의 정수 N과, N개 작업의 작업 명, 시작 시간, 종료 시간, 작업 대금(수익)을 차례로 입력받아 구조체 배열에 저장하고, 작업 시간이 겹치지 않는 작업 3개를 검색하여 그 중 가장 수익이 많게 되는 작업 3개의 정보를 출력하는 프로그램을 작성하시오. 만일 같은 수익을 내는 경우가 있으면 먼저 입력된 작업이 속한 경우를 선택한다. 출력 할 때 작업명의 순서는 입력 순서에 따라 출력한다.

입력 정보 : 작업의 개수 N과, N개의 작업 (아래의 1~4 항목) 정보 (N >= 3)

1. 작업명 : 영문 최대 100자
2. 작업 시작 시간 : "HH:MM:SS" 형태로 24시간 제 표기를 따른다. (시:분:초)
3. 작업 종료 시간 : 위와 동일
4. 수익금 : 작업에 대한 수익

출력정보 : 1. 수익이 최대가 되는 작업 3개의 작업명 출력 (작업 시간이 겹치지 않으면서)
사용자의 입력순서에 따라 3개 작업명 출력한다.
2. 수익금

3. 3가지 작업의 전체 시작시간/종료시간

(주의: 3개 작업의 수익이 최대인 경우가 여러 개이면, 첫 작업의 사용자 입력 순서가 빠른 3개 작업 선택, 첫 작업이 동일하면, 두 번째 작업의 사용자 입력순서가 빠른 3개 작업 선택, 앞의 두 작업이 동일하면, 세 번째 작업의 사용자 입력순서가 빠른 3개 작업 선택)

1) 작업 목록을 저장하기 위하여 아래 구조체 자료형을 선언하여 사용한다.

```
struct job{
   char *name;      // 작업명 (공백없이 최대 100자)
   char *start;     // 작업 시작 시간 (HH:MM:SS; 24시간제 시간,분,초 예)15:32:35)
   char *end;       // 작업  종료 시간 (위와 동일)
   int money;       // 작업 대금 (수익)
} ;
```

- typedef를 사용하여 job 구조체의 type명을 ABC로 바꾸어 사용하시오.
- 구조체 멤버를 추가하지 마시오.

2) 구조체 내부의 name은 공백 없이 최대 100자인 문자로, 메모리를 절약하기 위해
문자배열을 동적 할당하여 사용하고 start, end는 24시간제 표기를 담을 수 있는 만큼
문자배열 동적 할당하여 사용. (예: 15:32:35 – 오후 3시 32분 35초) 예: 00:00:01
멤버 변수 name, start, end에 메모리 동적할당하여 연결할 때 정확한 크기로 할당하시오.

3) 작업 3개를 선정할 때 서로 작업시간이 겹치지 않아야 한다.
종료시간과 시작시간이 같은 것은 가능하다. 작업1이 13:00:00 에 끝나고 작업2가 13:00:00 에 시작한다면 서로 작업시간이 겹치지 않는다.
(힌트) 작업 3개씩 선정하여, check_dup 함수 호출하여, 작업이 겹치지 않는 경우를 찾는다.

4) 다음의 2가지 함수를 사용하여 작업한다.

함수 1: 작업 3개의 정보를 받아 시간이 겹치는 지 여부를 체크하여 반환하는 함수

함수원형 : int **check_dup**(struct job job1, struct job job2, struct job job3);

함수 인자 : 작업 구조체 원소 3개

반환값 : 1 – 작업시간이 겹치는 경우

0 – 작업시간이 겹치지 않는 경우

함수 2: 작업 3개 전체의 시작시간과 종료시간을 구하는 함수. (전체 작업 시작/종료시간)

함수 원형 : char ***get_start_end**(struct job job1, struct job job2, struct job job3)

함수 인자 : 작업 구조체 원소 3개

반환값 : 시작 시간과 종료 시간을 담은 문자열의 주소

하는 일 : job1~job3의 시작시간 중 제일 빠른 것과 종료시간 중 제일 느린 것을 합쳐서 하나의 문자열에 저장하고 이의 주소를 반환한다.

예) "13:00:00 15:00:00"
문자열을 담을 메모리를 함수 내에서 동적 할당. 메인에서 동적할당 해제

5) 3가지 작업을 선택할 수 없는 경우 (즉, 시간이 안 겹치는 3개 작업이 없는 경우) NONE을 출력한다. (입력예시3)

입력 예시 1

```
4          ↦ 작업 개수
BasicPaint 09:45:00 10:30:00 10
Drawing 08:27:45 13:10:00 20
Paint 11:00:00 14:00:00 40
Remove 14:00:15 16:00:00 30
```

출력 예시 1

```
BasicPaint Paint Remove
80
09:45:00 16:00:00
```

입력 예시 2

```
7          ↦ 작업 개수
BasicPaint 09:45:00 10:30:00 50
Drawing 08:27:45 13:10:00 50
Paint 09:27:45 10:00:00 50
Remove 13:00:15 14:00:00 100
Cleaning 13:00:15 18:00:00 100
Finish1 15:00:00 16:00:00 10
Waste 16:00:00 18:00:00 30
```

출력 예시 2

```
BasicPaint Remove Waste
180
09:45:00 18:00:00
```

수익이 가장 큰 3개씩 작업 BasicPaint Remove Waste (180)과 Paint Remove Waste(180) 중 첫 작업의 입력 순서가 빠른 BasicPaint Remove Waste 출력

입력 예시 3

```
4    ↦ 작업 개수
BasicPaint 09:45:00 16:30:00 50
Drawing 10:33:45 16:10:00 50
Paint 13:27:45 16:10:00 50
Remove 13:00:15 16:10:00 100
```

출력 예시 3

```
NONE
```

- 구조체 배열 동적할당 사용하지 않은 경우 (100% 감점)
- 함수 사용 안하거나 원형이 틀린 경우 (각 30% 감점)
- 메모리 동적 할당 오류체크 한 번도 하지 않았으면 (최소 1회 시행) (20% 감점)
- get_start_end안에서 동적 할당 한 메모리를 메인에서 해제 안하면 (20% 감점)
- 구조체 배열/멤버변수들의 메모리 해제 안하고 종료하는 경우가 발생하면
  - 코드에 표기는 있으나 실행이 안 되는 경우 포함 (20% 감점 – 부분 점수 없음)
- 구조체 멤버 변수들 동적 할당 안하면 (name 20%감점, start/end 각각 10%감점)
- 구조체 멤버 변수들 동적 할당 할 때 크기 틀리면 (name/start/end 각각 5%감점)
- 구조체 멤버 추가하면 10% 감점
- typedef 사용하지 않으면 10% 감점