Problem 1.

We want to build a program counting the number of words in a text file. Check the main() function in the given source code that opens a text file and reads the contents of the file into a dynamically allocated memory.

You need to complete the **count_words()** counting the number of words in a character array pointed by the "txt". "txt" is pointing to the dynamically allocated memory used to read in the text file.

Note that the character array may include multiple strings separated by a new line character, '\n\r'. A null character, '\0' exits at the end of the character array.

You can freely use standard library functions like strtok() if necessary. You can also add your own functions.

(텍스트 파일의 단어 수를 세는 프로그램을 작성하고자 한다. 주어진 소스 코드의 main() 함수를 확인하라. main() 함수는 텍스트 파일을 열고 그 내용을 동적으로 할당된 메모리에 저장한다. 당신이 할 일은 "txt"가 가리키는 문자배열의 단어수를 세는 count_words()를 완성하는 것이다. "txt"는 파일의 내용을 읽어 들이기 위해 동적으로 할당된 메모리를 가리키고 있다. 이 배열 안에는 '\n\r'으로 나뉘는 여러 문자열이 있을 수 있다. 문자 배열의 끝에는 '\0'이 존재한다. strtok()와 같은 라이브러리 함수를 사용할 수 있고 함수를 추가하여도 된다)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>

int count_words(char *txt) {
// Enter your code

  return 0;
}

int main(void) {
  char * filename = "data/ai.txt";
  FILE *fd = NULL;
  int test_case = 1;

  scanf("%d", &test_case);

  switch (test_case) {
    case 2:
      filename = "data/internet.txt";
      break;
    case 3:
      filename = "data/ce.txt";
      break;
    default :
      break;
  }
```

```
    fd = fopen(filename, "rt");
    if (fd) {
        int nsize;
        char * buffer = NULL;

        fseek(fd, 0, SEEK_END);
        nsize = ftell(fd);
        rewind(fd);
```

```
        buffer = (char *)malloc(nsize+1);
        assert(buffer);

        nsize = fread(buffer, sizeof(char), nsize, fd);
        buffer[nsize] = '\0';
        printf("# of words in %s is %d\n",filename,count_words(buffer));
        fclose(fd);
    }
    else puts("File Open Error\n");

    return 0;
}
/*
// ai.txt
Artificial intelligence (AI), is intelligence demonstrated by machines, unlike the natural intelligence displayed by humans and animals.

// internet.txt
The Internet is the global system of interconnected computer networks that uses the Internet protocol suite to communicate between networks and devices. It is a network of networks of local to global scope, linked by a broad array of electronic, wireless, and optical networking technologies.
*/

```

```c
/*
// ce.txt
Computer engineering is a branch of engineering that integrates several fields of computer science and electronic engineering required to develop computer hardware and software. Computer engineers usually have training in electronic engineering (or electrical engineering), software design, and hardware-software integration instead of only software engineering or electronic engineering. Computer engineers are involved in many hardware and software aspects of computing, from the design of individual microcontrollers, microprocessors, personal computers, and supercomputers, to circuit design. This field of engineering not only focuses on how computer systems themselves work but also how they integrate into the larger picture.
Usual tasks involving computer engineers include writing software and firmware for embedded microcontrollers, designing VLSI chips, designing analog sensors, designing mixed signal circuit boards, and designing operating systems. Computer engineers are also suited for robotics research, which relies heavily on using digital systems to control and monitor electrical systems like motors, communications, and sensors.
*/
```

Problem 2.

Complete the **sort_students(STUD *ps)** that sorts an array of STUD types whose address is given as "ps" in the **decreasing** order of points. The code will use the sorted STUD array to call the print_tops() that prints out all items having the highest point.

You can add your own functions freely if necessary.

(STUD 구조체 배열을 points 값에 대해 내림차순으로 정렬하는 sort_students() 함수를 완성하라. 정렬된 배열은 print_tops()라는 함수 호출에 쓰이는데, print_tops()는 가장 높은 point를 가지는 모든 항목을 출력한다. 함수 구현에 필요하면 새로운 함수를 추가하여도 된다)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>

typedef struct student {
  int id;
  char name[12];
  double points;
} STUD;

STUD * sort_students(STUD *ps) {

  return ps;
}

void stud_print(STUD *ps) {
  printf("%03d %-12s %5.1lf\n", ps->id, ps->name, ps->points);
}

void print_students(STUD *ps) {
  while (ps->id > 0) stud_print(ps++);
}
```

```c

void print_tops(STUD *sorted_ps) {
  double highest_points = sorted_ps->points;
  do {
    stud_print(sorted_ps++);
  } while (sorted_ps->points == highest_points);
}

int main(void) {
  STUD test_data[] = { {1, "Choi", 9.9}, {2, "Park", 0.1}, {3, "Kim", 5.0 },
  {4, "Lee", 9.9 }, {5, "Moon", 9.5 }, {6, "Kang", 7.0 }, {7, "Jeon", 0.9 },
{-1, "", 0 } };
  STUD *students = test_data, *sorted_students = NULL;
  int test_case = 1;

  scanf("%d", &test_case);

  sorted_students = sort_students(students);
  print_tops(sorted_students);

  return 0;
}
```

Problem 3.

Complete the **read_students_text(STUD *ps)** and **read_students_binary(STUD *ps)** in the given source code reading STUD type values from the given file and returning the pointer of a STUD array containing the read values. The STUD array returned by the functions needs to be allocated dynamically inside the functions. To indicate the end of the array, append an item having a negative "id" value after valid items.

(주어진 파일에서 STUD 구조체 자료 값들을 읽어 들이는 read_students_text(), read_students_binary() 함수를 완성하라. 이 함수는 읽어 들인 STUD 자료 값들을 모두 포함하고 있는 STUD 배열의 포인터를 반환하여야 한다. 값을 저장하는 STUD 배열은 함수 내에서 동적으로 할당되어야 한다. 그리고 배열의 끝은 표시하기 위해 음의 "id" 값을 가지는 항목을 마지막에 추가하라.)

The file to be read by "read_students_binary()" is written through the function **"save_students_binary()"** included in the given source code. Similarly, the file to be read by"read_students_text()" is written through the function **"save_students_text()"**. To verify your function, "print_tops()" function printing out all items having the highest point value will be called.

(read_students_binary() 가 읽을 파일은 save_students_binary()라는 함수에 의해 만들어졌으며 read_students_text()는 save_students_text()라는 함수에 의해 만들어졌다. 이들 함수의 코드는 주어진 코드에 포함되어 있다. 구현한 함수가 제대로 값들을 읽어 오는 지 간단히 확인하기 위하여 가장 높은 점수를 가진 모든 항목을 출력하는 print_tops()가 호출된다.)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>

typedef struct student {
  int id;
  char name[12];
  double points;
} STUD;

STUD * read_students_text(char *filename) {

  return NULL;
}

STUD * read_students_binary(char *filename) {

  return NULL;
}
```

```c
void save_students_text(STUD *ps, char *filename) {
  FILE *fd = fopen(filename, "wt");

  if (fd) {
    while (ps->id > 0) {
      fprintf(fd, "%d %s %lf\n", ps->id, ps->name, ps->points);
      ps++;
    }
    fclose(fd);
  }
}

void save_students_binary(STUD *ps, char *filename) {
  FILE *fd = fopen(filename, "wb");
  STUD *pcur = ps;
  int nitem = 0;
  if (fd) {
    while (pcur->id > 0) pcur++;
    fwrite(ps, sizeof(STUD), pcur - ps, fd);
    fclose(fd);
  }
}

void stud_print(STUD *ps) {
  printf("%03d %-12s %5.1lf\n", ps->id, ps->name, ps->points);
}

void print_students(STUD *ps) {
  while (ps->id > 0) stud_print(ps++);
}

void print_tops(STUD *sorted_ps) {
  double highest_points = sorted_ps->points;
  do {
    stud_print(sorted_ps++);
  } while (sorted_ps->points == highest_points);
}
```

```c
STUD * sort_students(STUD *ps);

int main(void) {
  char *filename = "data/output";
  STUD test_data[] = { {1, "Choi", 9.9}, {2, "Park", 0.1}, {3, "Kim", 5.0 },
  {4, "Lee", 9.9 }, {5, "Moon", 9.5 }, {6, "Kang", 7.0 }, {7, "Jeon", 0.9 },
{-1, "", 0 } };
  STUD *students = test_data, *readin_students = NULL, *sorted_students =
NULL;
  int test_case = 1, bbinary = 0;

  scanf("%d", &test_case);
    if (bbinary) {
    save_students_binary(students, filename);
    readin_students = read_students_binary(filename);
  }
  else {
    save_students_text(students, filename);
    readin_students = read_students_text(filename);
  }
  if (readin_students) {
    sorted_students = sort_students(readin_students);
    print_tops(sorted_students);
  }
  return 0;
}
```