프로그래밍 과제 03

1. `rects.txt` 파일에는 사각형들에 대한 정보가 저장되어 있다. 파일의 첫 줄에는 사각형의 개수 N이 주어지고, 이어진 N줄에는 각 줄 마다 하나의 사각형에 대한 정보가 주어진다. 하나의 사각형은 왼쪽-아래쪽 꼭지점의 x-좌표와 y-좌표, 사각형의 너비와 높이의 4개의 정수로 표현되어 있다. (왼쪽-아래쪽 꼭지점은 사각형의 네 꼭지점 중에서 x-좌표와 y-좌표가 최소인 꼭지점을 의미한다.)

(1) 먼저 파일에 저장된 사각형의 정보를 읽어서 파일에 저장된 순서대로 저장하는 연결리스트를 구성한다. 연결리스트에 저장된 순서대로 출력한다.

(2) 사각형들을 면적에 관한 오름차순으로 정렬하고 다시 출력한다. 정렬을 할 때는 실제로 노드들이 연결된 순서를 변경해야 하며, 노드에 저장된 데이터만 복사하는 방식으로 해서는 안된다.

(3) 두 정수 `min_w`와 `min_h`를 키보드로 부터 입력받은 후 너비가 `min_w` 미만이거나 높이가 `min_h` 미만인 사각형들을 모두 찾아 연결리스트로부터 삭제한다.

프로그램의 기본 틀은 아래에 주어진 것을 수정없이 사용해야 한다.

```
struct Node {
    int x, y, w, h;
    Node *next;
}

Node *head = nullptr;

void print_list()
{
    Node *p = head;
    while(p!=nullptr) {
       cout << p->x << " " << p->y << " " << p->w << " " << p->h << endl;
       p = p->next;
    }
}

void read_file()
{
    // rects.txt 파일을 읽어서 사각형들을 파일에 저장된 순서대로
    // head가 가리키는 연결리스트에 저장한다.
}

void sort_by_area() {
    // head가 가리키는 연결리스트를 면적순으로 정렬한다.
}

int main()
{
    // (1)
    read_file();   // 파일을 읽어서 파일에 저장된 순서대로 저장된 연결리스트를 구성한다.
    print_list();  // 파일에 저장된 순서대로 출력된다.

    cout << endl;  // 한 줄을 띄운다.

    // (2)
    sort_by_area();   // 연결리스트의 노드들을 면적순으로 정렬한다.
    print_list();     // 정렬된 순서대로 출력한다.
```

```
        cout << endl;  // 한 줄을 띄운다.

        // (3)
        int min_w, min_h;
        cin >> min_w >> min_h;
        remove_rects(min_w, min_h);
        print_list();

        return 0;
    }
```

입력 파일 RECTS.TXT의 예	출력 예

16
2 3 1 12
0 0 4 5
2 2 8 4
-2 -4 2 2
1 4 4 2
-4 2 5 5
0 0 8 5
3 7 4 6
5 2 7 6
0 3 2 8
1 4 1 1
4 7 7 3
6 1 4 9
2 7 18 1
9 4 3 6
0 6 6 2
 	
// 키보드입력
w_min = 3
h_min = 2
	2 3 1 12
	0 0 4 5
	2 2 8 4
	-2 -4 2 2
	1 4 4 2
	-4 2 5 5
	0 0 8 5
	3 7 4 6
	5 2 7 6
	0 3 2 8
	1 4 1 1
	4 7 7 3
	6 1 4 9
	2 7 18 1
	9 4 3 6
	0 6 6 2
	
	1 4 1 1
	-2 -4 2 2
	1 4 4 2
	2 3 1 12
	0 6 6 2
	0 3 2 8
	2 7 18 1
	9 4 3 6
	0 0 4 5
	4 7 7 3
	3 7 4 6
	-4 2 5 5
	2 2 8 4
	6 1 4 9
	0 0 8 5
	5 2 7 6
	
	1 4 4 2
	0 6 6 2
	9 4 3 6
	0 0 4 5
	4 7 7 3
	3 7 4 6
	-4 2 5 5
	2 2 8 4
	6 1 4 9
	0 0 8 5
	5 2 7 6

2. 입력으로 텍스트 파일 harry.txt를 읽는다. 이 텍스트 파일은 오직 영문 소문자만으로 구성되어 있다. 이 파일에서 다음과 같은 일을 순서대로 수행하는 프로그램을 작성하라(각각을 별개의 프로그램으로 만들지 말고 하나의 프로그램으로 구현하라.)

(1) 입력 파일에 등장하는 모든 단어의 목록과 각 단어의 등장 빈도를 구하여 화면으로 출력한다. 단어들은 사전식 순서로 정렬되어 출력되어야 한다. 출력 파일의 각 줄에 하나의 단어와 그 단어의 등장 빈도를 콜론(:)으로 구분하여 출력하라. 동일한 단어가 중복해서 출력되어서는 안된다. 출력의 마지막에는 전체 단어의 개수를 출력하라. 아래는 올바른 출력의 시작 부분과 끝 부분을 보여준다. **단어들은 하나의 연결리스트로 저장되어야** 한다. **주의:** 파일로부터 단어들을 읽어서 먼저 배열에 모두 저장한 후 나중에 다시 연결리스트 리스트로 만드는 식으로 해서는 안된다. 단어들은 파일에서 읽는 즉시 연결리스트에 저장되어야 한다. 또한 중복 검사 없이 모든 단어를 연결리스트에 저장한 후 나중에 중복을 제거하는 식으로 해서도 안된다. 연결리스트에는 항상 단어들이 중복없이 저장되어 있도록 구현해야 한다.

```
a: 478
abandoning: 1
able: 6
ably: 1
about: 52
above: 2
abroad: 1
absence: 1
accepting: 1
accidental: 2
accommodate: 1
…
younger: 1
your: 31
yours: 2
youthis: 1
zards: 1
zigzagged: 1
zoomed: 4
zooming: 2
2763
```

(2) (1)번에서 만든 연결리스트에서 등장 빈도가 10 이하인 모든 단어들을 연결리스트로부터 제거한 후 남아 있는 단어들만 1번과 같은 형식으로 출력하라. 단어들은 사전식 순서로 정렬되어 있어야 하며, 마지막에 남아있는 단어들의 개수를 출력한다.

```
a: 478
about: 52
across: 14
all: 70
an: 45
and: 552
any: 20
anything: 16
are: 30
…
without: 11
wizard: 16
wizards: 15
world: 24
would: 42
you: 136
your: 31
```

```
211
```

(3) (2)번에서 만든 연결리스트에서 노드들을 단어들의 등장 빈도의 내림차순으로 정렬하라. 등장 빈도가 동일한 단어들은 사전식 순서로 정렬되어야 한다. (2)번에서 만든 연결리스트에서 노드들을 하나씩 떼어서 새로운 정렬된 연결리스트에 삽입하는 방식으로 구현하라. 정렬된 연결리스트를 (1)번과 (2)번과 같은 형식으로 출력하라.

```
the: 992
and: 552
to: 519
a: 478
of: 429
he: 347
his: 294
was: 290
said: 268
had: 265
…
owl: 11
people: 11
school: 11
set: 11
three: 11
top: 11
while: 11
without: 11
211
```

3. 수업에서 다른 다항식 프로그램에 다음 예와 같이 두 다항식을 더해서 새로운 다항식을 정의하는 명령 `addpoly`와 두 다항식을 곱해서 새로운 다항식을 정의하는 명령 `multiplypoly`를 추가하라.

```
$ define f
$ add f -8 8
$ add f -2 0
$ add f 6 2
$ define g
$ add g 8 8
$ add g 4 2
$ addpoly h f g                // h = f + g
$ print h
h=10x^2-2
$ multiplypoly k g h           // k = g * h
$ print k
k=80x^10-16x^8+40x^4-8x^2
```

4. 다음의 프로그램을 완성하라. 이 프로그램은 n개의 단어를 입력받아서 사전식 순서로 2중 연결리스트에 저장한 후 중복된 단어를 찾아서 제거하는 일을 한다.

```
struct Node {
    string data;
    Node *prev, *next;
};
```

```
void ordered_insert(string item);
void remove_dup();
void print_list_twice();

Node *head = nullptr, *tail = nullptr;  /* 2중 연결리스트의 처음과 마지막 노드 */

int main() {
    int n;
    string word;
    cin >> n;
    for (int i=0; i<n; i++) {
        cin >> word;
        ordered_insert(word);
    }

    print_list_twice();
    remove_dup();
    print_list_twice();
    return 0;
}

void ordered_insert(string item)
{
  /* head와 tail이 가리키는 2중 연결리스트에 문자열들이 사전식 순서로 정렬된 순서를 유지하
     도록 새로운 문자열 item을 삽입한다. */
}

void remove_dup()
{
  /* 2중 연결리스트에 저정된 문자열들 중에서 모든 중복된 문자열을 찾아 하나만 남기고 제거한
     다. */
}

void print_list_twice()
{
    Node *p = head;
    while(p != nullptr) {
        cout << p->data << " ";
        p = p->next;
    }
    cout << endl;

    Node *q = tail;
    while(q != nullptr) {
        cout << q->data << " ";
        q = q->prev;
    }
    cout << endl;
}
```

가령 $n = 11$이고, 입력된 단어들이 다음과 같다면

```
coding is so fun but coding is also so so painful
```

이 프로그램의 출력은 다음과 같아야 한다.

```
also but coding coding fun is is painful so so so
so so so painful is is fun coding coding but also
also but coding fun is painful so
```

```
so painful is fun coding but also
```

함수 `ordered_insert`와 `remove_dup`을 완성하는 것 이외에 프로그램의 다른 부분을 변경하거나 전역변수 혹은 다른 함수를 추가해서는 안된다. Garbage 문제는 고려하지 않아도 된다.