고급객체지향프로그래밍 과제 #2

강대기

2008년 4월 4일

제 1 절 과제 제출에 대해 반드시 알아야 할 사항

본 과제의 데드라인은 4월 20일 일요일 밤 11시 59분이다.
본 과제는 다음 두가지 방법 중 하나로 제출될 수 있다.

1. 우선 이메일로 제출하는 방법이 있다.

   과제를 제출할 때는 프로그램의 소스 파일들을 ZIP 으로 압축하여 dkkang@dongseo.ac.kr로 이메일로 보낸다.

   이메일로 제출할 때, 이메일의 제목을 "[OOP2008]"로 시작하고 "HW2", 반, 학번, 이름 등을 순서대로 기재한다.

   예를 들어 E11반 학번 20061111인 홍길동이 과제 #2을 제출하는 경우 다음과 같이 이메일 제목을 쓰면 된다.

   ```
   제목: [OOP2008]HW2.E11.20061111.홍길동
   ```

   반드시 그렇게 안해도 되지만, 가능하면 파일명도 이메일 제목과 일치시키면 강사가 과제를 분류하는 수고를 덜 수 있다. 즉 위와 같은 경우, `[OOP2008]HW2.E11.20061111.홍길동.zip` 이라고 파일명을 만들면 받는 쪽에서 더 좋다는 것이다.

   이메일의 제목을 저렇게 써주면 실제로 본 강사의 컴퓨터에서는 메일 클라이언트가 이메일을 자동으로 분류한다. 그런데, 이렇게 적어도 실제로 과제를 받아보면 건성으로 제출 사항을 읽고 틀리게 보내는 학생들이 대다수이니, 조그만 더 신경 써주었으면 하는 바램이다.

   저런 규칙을 안지키는 건 대부분의 경우 조금 불편한 정도지만, 가장 최악인건 보내는 학생의 이름을 쓰지 않는 경우이다. 초기에 자신의 이메일 설정에서 자신의 본명을 넣지 않는 경우가 대부분인데, 과제를 제출하는 이메일 제목이나 본문에도 이름을 넣지 않으면 받는 사람이 친한 사람이 아닌 이상 상대가 누군지 알 수 없다. 그런데도, 실제로 이름도 안쓰고 저렇게 하지도 않고 그냥 보내는 학생들도 더러 있어 왔다. 강사로선 가능한 모든 방법을 동원해서 이름을 유추해 왔는 데 그게 불가능하거나 상당히 시간을 소모한다.

   따라서 앞으로는, 이메일 제목이나 본문에 자신의 이름을 넣지 않는 학생은, 시험지에 본인의 이름을 안쓴 거처럼, 당연히 점수를 줄 수 없다는 점을 명확히 하고 싶다.

2. 첫번째 제출 방법은 웹 버그(Web bug)를 통해 상대방이 이메일을 봤는지 확인하는 경우 마음은 놓이지만, 강사의 메일 클라이언트의 경우 그런 확인 방법을 기본적으로 미리 막고 있다. 또한 자신의 이메일이 스팸으로 메일이 분류될 수도 있으므로, 학생들에 따라서는 마음이 놓이지 않을 수도 있을 것이다.

   두번째 제출 방법은 이번 학기부터 수행되는 웹 클래스에 의한 제출방법이다. 웹 클래스의 홈페이지는 `http://webclass.dctl.ac.kr/`이다. 웹 클래스에 들어가서 자신의 과목과 반에 들어가면 과제를 제출할 수 있게 되어 있다. 대다수의 학생들이 웹 클래스를 사용해 봤으므로 이 방법에 대해 더 길게 설명할 필요가 없을 것이다.

다시 한번 말하지만 위의 두가지 방법 모두 써서 제출해도 되고, 그 중 한가지 편한 방법을 골라서 제출해도 된다. 다만, 두가지 방법을 모두 쓸 경우, 강사가 알아서 하나를 선택해서 채점할 것이다.

본 과제를 풀 때, 학생들 간에 문제를 풀기 위한 토의는 허용된다. 즉 학생들은 서로 문제를 어떻게 풀 것인가에 대해 아이디어를 교환할 수 있으며, 이 때, 노트나 칠판을 사용할 수 있다. 또한 문제를 풀기 위해 인터넷을 참고할 수 있다.

그러나, 학생들이 문제에 대해 실제 교수에게 제출할 답안을 작성할 때에는 웹 브라우저를 반드시 끄고 비주얼 C++와 텍스트 에디터 또는 MS 워드나 한글을 가지고 본인 스스로 답안을 작성해서 보내야 한다. 또한 다른 학생에게 아이디어를 얻었을 경우, 각각의 과제의 답안 뒤에 그 학생의 이름을 반드시 언급해야 한다. 예를 들어 "홍길순에게 본 답안의 아이디어를 얻었습니다"라고 써서 제출한다. 이러한 경우, 감점은 전혀 없다.

만일 인터넷에서 답을 본 경우, 그 답을 머리로 이해한 후, 웹 브라우저를 끄고, 혼자 힘으로 그 답안을 작성한다. 그리고 나서 마지막에 참고 문헌으로 인터넷의 URL 주소와 페이지의 제목을 적어서 제출한다. 이러한 경우에도, 감점은 전혀 없다.

만일 그렇지 않은 경우, 내용이 비슷한 답안이 적발되면 적발된 모든 학생에 대해 예외없이 0점 처리를 한다. 원래의 답을 쓴 사람도 0점이고, 베껴쓴 사람도 0점이다. 학생은 자신의 답을 남이 함부로 훔쳐서 보지 못하도록 간수할 책임이 있다.

만일 자신이 비주얼 C++로 프로그램을 작성할 때 옆에서 다른 학생이 도와줘서는 안되며, 다른 학생이 프로그램을 작성할 때 도움을 주어서도 안된다.

제 2 절 과제

본 과제는 10 문제이다. 처음 다섯 문제는 객체 지향에 대한 지식을 테스트하고, 다른 다섯 문제는 기초적인 알고리즘에 대한 기초적인 매우 기초적인 지식을 테스트한다.

2.1 객체는 초기화해 드리는 게 인지상정

다음과 같은 프로그램을 보자.

```
01: #include <iostream>
02:
03: int main() /* main */
04: {
05:   std::cout << "이 세계의 파괴를 막기 위해" << std::endl;
06:   std::cout << "이 세계의 평화를 지키기 위해" << std::endl;
07:   return 0;
08: }
```

이 프로그램을 변경하여, 다음과 같이 출력되도록 해보자.

```
우리가 누군지 물으신다면
대답해 드리는 게 인지상정.
이 세계의 파괴를 막기 위해
이 세계의 평화를 지키기 위해
사랑과 진실, 어둠을 뿌리고 다니는
포켓몬의 감초, 귀염둥이 악당
```

단, 저 위의 main 함수는 그 내용이 절대 바뀌면 안되고 그대로이어야 한다. 제출할 프로그램의 소스 코드의 이름은 `pokemon.cpp`로 하면 된다.

2.2 double과 객체의 곱셈

11 장에서 배운 Stonewt 객체를 생각해 보자.

1. double 과 Stonewt 객체의 곱셈, 그리고 Stonewt 객체들 간의 곱셈을 멤버 함수와 프렌드 함수를 이용하여 구현하라.

   클래스 이름은 Stonewt1 이라 하고 제출할 프로그램의 소스 코드의 이름은 `Stonewt1.h`와 `Stonewt1.cpp`로 한다.

2. 프렌드 함수들만 가지고 연산자 오버로딩하여 double 과 객체의 곱셈, 그리고 객체들 간의 곱셈을 구현하라.

   클래스 이름은 Stonewt2 이라 하고 제출할 프로그램의 소스 코드의 이름은 `Stonewt2.h`와 `Stonewt2.cpp`로 한다.

3. 프렌드 함수와 생성자를 통한 형 변환을 이용하여 double 과 객체의 곱셈, 그리고 객체들 간의 곱셈을 구현하라.

   클래스 이름은 Stonewt3 이라 하고 제출할 프로그램의 소스 코드의 이름은 `Stonewt3.h`와 `Stonewt3.cpp`로 한다.

2.3 INT

정수 형인 int 하고 똑같이 작동되는 클래스 INT 를 정의하고 구현하라. 이를 위해서는, 더하기, 빼기, 곱하기, 나누기 등등의, 지난 시간에 연산자 오버로딩에서 배운 교재의 표(645, 646쪽)에 나오는 연산자들 중에서 정수 형이 적용되는 모든 연산자 오버로딩은 물론, 변환 함수들까지 사용해야 한다. 다시 말하지만 단순히 사칙연산만 하지 말고 정수 형이 적용될 수 있는 가능한 모든 연산자들에 대해 오버로딩을 해보자.

제출할 프로그램의 소스 코드의 이름은 `INT.h`와 `INT.cpp`로 한다.

2.4 LLLINT

이번에는 임의의 거대한 수의 연산도 가능한 정수 클래스를 구현해 보자.

정수는 보통 32비트 또는 64비트, 128비트 정도의 크기를 가지는 데, 아예 임의의 아주 큰 수의 연산도 가능하게 해보자는 것이다.

이를 위해 우선 다음 두 가지 경우를 나누어서 문제를 풀어보자.

1. 일단 객체 지향은 생각하지 말고, 1000 의 팩토리얼 (1000!)을 구해보자. 이를 위해 어떻게 덧셈과 곱셈 연산을 정의하고, 어떻게 숫자들을 저장해야 할지 고민해서 프로그램을 만들도록 한다. 이 프로그램은 실행하면 1000!의 결과로 나오는 모든 자리의 숫자를 정확히 출력해야 한다. 예를 들면 `1.234E+567` 이라는 식의 지수 표현으로 출력되선 안된다.

   제출할 프로그램의 소스 코드의 이름은 `1000.cpp`로 한다.

2. 임의의 거대한 수의 연산이 가능한 정수 클래스의 이름을 LLLINT 라고 하고 실제로 구현해 보자. 이를 위해서는, 더하기, 빼기, 곱하기, 나누기 등등의 지난 시간에 연산자 오버로딩에서 배운 교재의 표(645, 646쪽)에 나오는 것들 중 가능한 모든 연산자들을 오버로딩해야 한다.

   제출할 프로그램의 소스 코드의 이름은 `LLLINT.h`와 `LLLINT.cpp`로 한다.

2.5 String의 substring

12 장에서는 동적 메모리 할당을 제대로 고려하여 String 객체를 구현하기 위한 방법도 배웠다.

1. 이제 substring 기능을 `()` 연산자를 오버로딩하는 방법으로 구현하라. 즉, 다음과 같은 코드들이 가능하도록 하는 것이다.

   ```
   String str1("Hello");
   String str2("World");

   String str3 = str1(2); // str3 == "llo"
   String str4 = str2(1,4); // str4 == "orl"
   ```

   제출할 프로그램의 소스 코드의 이름은 `String1.h`와 `String1.cpp`로 한다.

2. 이제 프로그램을 더 다듬어서, substring 기능이 대입 연산자의 왼쪽에도 올 수 있도록 한다. 예를 들어 다음과 같은 코드들이 가능하도록 하자는 것이다. [1]

```
String str1("Hello");
String str2("World");

str1(1) = "elp!"; // str1 == "Help!"
str2(1,4) = "ooo"; // str2 == "Woood"
str1(1) = "appiness"; // str1 == "Happiness"
str2(1,4) = "oo"; // str2 == "Wood"
```

제출할 프로그램의 소스 코드의 이름은 `String2.h`와 `String2.cpp`로 한다.

2.6 RLE(Run Length Encoding)

다음은 게임 회사 넥슨의 입사 문제로 인터넷에 공개된 문제 중 하나이다.

RLE(Run Length Encoding)란, 임의의 수열을 (반복 수, 숫자)의 쌍으로 된 수열로 만드는 부호화 방법이다. 예를 들어 다음과 같은 열 개의 숫자로 된 수열이 있다고 할 때,

```
9, 9, 9, 5, 7, 7, 7, 7, 7, 7
```

RLE를 이용하여 다음과 같이 여섯 개의 숫자로 부호화할 수 있다.

```
(3,9), (1,5), (6,7)
```

해석하면, 9가 세 개 있고, 그 다음에 5가 한 개, 그 다음에 7이 여섯 개 나오는 수열이라는 뜻이다.

이 때, (원래 수열의 갯수) / (부호화 수열의 갯수) 를 압축률이라 하며, 위의 경우에서 압축률은 10 / 6 = 1.666 이다. 이렇게 부호화된 값은 쉽게 원래 값으로 복원할 수 있음을 알 수 있을 것이다.

그런데, 원래 값으로 되돌리는 것을 보장하지 않는 RLE 방법을 '손실 RLE'라 하자. 이 경우는 복원했을 때 오차가 생기게 되는데, 원래 수열과의 RMSE (Root Mean Square Error) 를 오차로서 정의한다. 크기가 n인 A 수열과 B 수열 사이의 RMSE는 다음과 같이 계산한다

$$RMSE(A,B) = \sqrt{\frac{(A1-B1)^2 + (A2-B2)^2 + ... + (An-Bn)^2}{n}}$$

'손실 RLE' 방법은 다양하게 존재하므로, 같은 수열이라도 다양하게 '손실 RLE' 부호화 할 수 있는데, 예를 들어, 위와 같은 수열의 경우 (10,9) 혹은, (10, 7) 혹은 (3, 9), (7, 7) 등으로 '손실 RLE' 부호화할 수 있을 것이다.

만약 (10,9) 으로 부호화했다면, 압축률은 5 이며, RMSE는 2 이다.

[1] 오타를 찾아준 연재혁, 서영조 학생에게 감사한다.

(10,7) 로 부호화했다면, 압축률은 5 이며, RMSE는 1.26 이다.
(3,9), (7,7) 로 부호화했다면, 압축률은 2.5 이며, RMSE는 0.63 이다.
이제 본격적으로 문제를 보도록 하자.
다음과 같이 128개의 정수로 된 수열이 있을 때,

```
49, 49, 50, 52, 49, 47, 47, 46, 44, 42, 42, 38, 38, 38, 36, 34,
33, 33, 33, 32, 34, 38, 42, 41, 42, 42, 40, 41, 40, 38, 38, 37,
37, 39, 41, 40, 40, 40, 40, 42, 45, 47, 47, 46, 46, 46, 47, 47,
47, 47, 46, 44, 43, 39, 36, 34, 34, 32, 30, 29, 30, 31, 31, 31,
30, 28, 25, 23, 24, 22, 22, 25, 25, 27, 31, 33, 35, 37, 38, 39,
39, 40, 40, 41, 40, 40, 40, 40, 39, 38, 37, 35, 35, 34, 33, 32,
31, 30, 30, 29, 29, 28, 27, 27, 28, 27, 25, 26, 0, 0, 90, 90,
90, 90, 90, 91, 90, 88, 87, 84, 80, 78, 83, 89, 91, 90, 89, 92
```

오차를 가능한 한, 작게 억제하면서 32개의 정수(압축률 4)로 압축하는 자신만의 '손실 RLE' 알고리듬을 만들고, 그 알고리듬에 의한 부호화 결과 수열과 RMSE를 적어라. (RMSE 가 1.8 미만이면 정답으로 간주하며, RMSE가 1.6보다 작을 수록 가산점 있음.)

주의할 점은 답은 정확히 32개의 정수로 나와야 하며 32개보다 적거나 많으면 오답으로 처리된다.

예를 들어, 다음은 정수의 개수가 모두 64 개 이므로 오답이다.

```
(8,49), (3,44), (5,38), (5,33), (1,38), (8,42), (6,38), (4,40),
(8,45), (4,47), (1,43), (1,39), (3,36), (2,32), (7,29), (3,25),
(3,22), (2,25), (2,31), (2,35), (5,38), (7,41), (3,37), (3,34),
(5,31), (7,28), (2,0), (9,90), (1,84), (2,80), (1,83), (5,89)
```

또한 다음은 정수의 개수가 30 개이므로 오답이다.

```
(8,47), (7,38), (6,33), (8,42), (11,40), (14,47), (12,31), (8, 24),
(20,39), (8,30), (6,26), (2,0), (9,90), (1,84), (8,83)
```

즉, 답은 다음과 같이 32 개의 정수와 1.8보다 작은 RMSE [2]가 나와야 한다.

```
(__,__),(__,__),(__,__),(__,__),(__,__),(__,__),(__,__),(__,__),
(__,__),(__,__),(__,__),(__,__),(__,__),(__,__),(__,__),(__,__)

RMSE = ____
```

2.7 Permutation 과 power set (멱집합)

1. 다섯 개의 숫자들을 입력 받아 가능한 모든 permutation 을 출력하는 프로그램을 작성하라. 출력될 행의 개수는 5! 즉 120 개 이다.

   예를 들어, 입력받은 숫자가 1,2,3,4,5 라면, permutation은 다음과 같을 것이다.

[2] 이 부분의 오류를 지적해 준 김현철 학생에게 감사한다.

```
1,2,3,4,5
1,2,3,5,4
...
5,4,3,1,2
5,4,3,2,1
```

프로그램의 이름은 `perm.cpp` 라고 한다.

2. 이번에는 다섯 개의 서로 다른 숫자를 입력받아 그 숫자들의 모든 부분 집합을 출력하는 프로그램을 작성하라. 출력될 행의 개수는 2^5 으로 32 개이다.

   예를 들어, 입력받은 숫자가 1,2,3,4,5 라면, 그 결과는 다음과 같을 것이다.

```
{}
{1}
{2}
...
{1,2,3,4}
{1,2,3,4,5}
```

   입력은 별도로 체크할 필요 없이, 서로 다른 다섯 개의 숫자라고 가정한다. 프로그램의 이름은 `power.cpp` 라고 한다.

2.8 Multiple links

Skip list 는 Binary search tree 의 대안 중 하나로, 상대적으로 단순한 구조를 가지고 있으며, 확률에 근거해 업데이트할 수도 있게 되어 있다 [1]. Skip list 는 몇몇 P2P 프로토콜에서 실제로 사용되고 있다 [2].

본 문제는 기존의 linked list 를 확장하여 skip list와 비슷한 구조를 만드는 것이다. 이를 위해 다음에 설명하는 바를 구현하여 `LinkedList.h` 와 `LinkedList.cpp` 라는 이름으로 제출한다.

일반적으로 linked list 는 다음과 같은 식으로 정의된다.

```
class LinkedList
{
  int value;
  class LinkedList* next; // 바로 다음 노드
};
```

Linked list에 대한 가능한 멤버 연산들은 다음이 있다.

```
LinkedList();
LinkedList(int value);
~LinkedList();
boolean insert(int index, int value);
boolean delete(int index, int& value);
boolean get(int index, int& value);
```

1. `LinkedList()` 는 디폴트 생성자로, 아무 데이터도 없는 비어있는 링크드 리스트를 만든다.

2. `LinkedList(int value)` 는 생성자로, value 값을 가지는 하나의 노드가 있는 링크드 리스트를 만든다.

3. `~LinkedList()` 는 파괴자이다.

4. `boolean insert(int index, int value)` 는 index 번째 위치에 새 노드를 추가한다. 새 노드의 값은 value 이다.

5. `boolean delete(int index, int& value)` 는 index 번째 위치의 노드를 없앤다. value 에는 없앤 노드의 값을 반환한다.

6. `boolean get(int index, int& value)` 는 index 번째 위치의 노드의 값을 value 에 반환한다.

7. insert, delete, get 함수는 index의 값이 범위 밖이거나 다른 이유로 함수의 수행이 실패하는 경우 false를 반환하고, 그렇지 않으면 true를 반환한다.

이제 이 링크드 리스트에 몇 개의 링크들을 더 추가해서 다음과 같은 구조로 확장하자.

```
class LinkedList
{
  int value;
  class LinkedList* next1; // 바로 다음 노드
  class LinkedList* next2; // 바로 다음의 다음 노드
  class LinkedList* next3; // 바로 다음의 다음의 다음 노드
  class LinkedList* next4; // 바로 다음의 다음의 다음의 다음 노드
};
```

위의 멤버 연산들도 이 자료 구조에 맞춰서 내부 구현을 바꾸도록 한다.

이제 이 링크드 리스트 구조에 대해서 << 연산자를 오버로딩하여 cout으로 출력할 수 있게 한다. 예를 들어 링크드 리스트에 1,2,3,4,5,6,7,8,9,10 이 들어 있다면, cout 으로의 출력 결과는 다음과 같이 나오도록 한다.

```
value = 1, next1 = 2, next2 = 3, next3 = 4, next4 = 5
value = 2, next1 = 3, next2 = 4, next3 = 5, next4 = 6
value = 3, next1 = 4, next2 = 5, next3 = 6, next4 = 7
value = 4, next1 = 5, next2 = 6, next3 = 7, next4 = 8
value = 5, next1 = 6, next2 = 7, next3 = 8, next4 = 9
value = 6, next1 = 7, next2 = 8, next3 = 9, next4 = 10
value = 7, next1 = 8, next2 = 9, next3 = 10
value = 8, next1 = 9, next2 = 10
value = 9, next1 = 10
value = 10
```

메인 프로그램에서는 이렇게 한다. 디폴트 생성자로 링크드 리스트를 하나 만들고, 난수를 20 개 발생하여 하나씩 링크드 리스트에 넣는다. 이렇게 한 후, 링크드 리스트를 cout에 출력한다. 그리고 나서, 링크드 리스트의 내용을 하나씩 지워서 전부 지운 뒤에 종료한다.

2.9 다항식 출력하기

8차 다항식을 표현하는 프로그램, `poly.cpp`를 만들어 보자.

각 항의 계수를 입력 받으면, 그 계수들에 따라 다항식을 표현한다. 8차 다항식이므로 9 개의 계수를 입력받아야 할 것이다.

다항식은 최대한 실제 수학에서 쓰이는 바에 가깝게 표현되도록 한다.

다음은 이를 위한 몇가지 원칙과 그 실행 예이다.

1. 당연히 차수가 높은 순서부터 낮은 순서로 출력된다.

   ```
   입력: 2 3 4 5 6 7 8 9 10
   출력: 2x^8 + 3x^7 + 4x^6 + 5x^5 + 6x^4 + 7x^3 + 8x^2 + 9x + 10
   ```

2. 차수는 ^ 뒤에 나타낸다.

3. 마지막에 나오는 상수는 상수 그대로 나타낸다. 즉 `10x^0` 으로 나타내지 말고 `10`으로 나타내야 한다.

4. 0 이 아닌 항만 출력한다.

   ```
   입력: 0 0 0 1 22 -333 0 1 1
   출력: x^5 + 22x^4 - 333x^3 + x + 1
   입력: 0 0 0 0 0 0 -55 5 0
   출력: -55x^2 + 5x
   ```

5. 각 항들 사이에 있는 더하기(+)와 빼기(-) 기호 양쪽으로만 공백을 넣는다. 그 외에는 출력에 공백이 있어선 안된다. 위의 실행 예에서 `입력`과 `출력`은 실제로 출력하지 않는다.

6. 맨 앞에 오는 항이 양수이면 부호를 출력하지 않고, 음수이면 `-7x^2 + 30x + 6` 과 같이 마이너스 부호를 출력한다.

7. 중간과 맨 뒤에 오는 음수 항은 "더하기 음수항" 이 아니라 "빼기 양수항" 으로 출력되야 한다. 즉 입력이 `0 0 0 0 0 0 1 -3 0` 이면 `x^2 - 3x`로 출력해야지, `x^2 + -3x`로 출력하면 안된다.

8. 항의 계수가 1이거나 -1이면 그에 맞게 출력되어야 한다. 예를 들어 다음과 같이 출력되어야 한다.

   ```
   입력: 0 0 0 0 0 -1 1 3 -1
   출력: -x^3 + x^2 + 3x - 1
   잘못된 출력: -1x^3 + 1x^2 + 3x - 1
   ```

2.10 시계의 바늘이 이루는 각도

만일 집의 벽에 큰 바늘과 작은 바늘을 가진 아날로그 시계가 있다면 유심히 살펴보기 바란다. 당연한 얘기지만 특정 시간에 대해 큰 바늘과 작은 바늘은 정확한 특정 각도를 이룬다. 또한 대부분의 제대로 만들어진 시계는, 시(時)를 가리키는 작은 바늘은 $60/5 = 12$ 분 마다 초침 하나, 즉 $360/60 = 6$도만큼 전진하고, 더 정확히는 2분마다 1도를 전진, 1분에는 0.5 도 전진한다 [3].

이제 이 사실을 염두에 두고, 특정 시각이 입력되면 그에 따른 큰 바늘과 작은 바늘의 각도를 출력하는 프로그램 `clock.cpp` 를 작성해 보자. 출력되는 각도는 0 도에서 180 도 까지의 각도이다. ($0 \leq \theta \leq 180$)

입력은 시(時)와 분(分)을 두 개의 숫자로 해서 하나의 시간을 받는다. 출력은 입력받은 시간으로 이루어지는 큰 바늘과 작은 바늘 간의 각도이다.

```
12 0
0.00

9 0
90.00

8 10
175.00
```

참고 문헌

[1] William Pugh. Skip lists: a probabilistic alternative to balanced trees. *Commun. ACM*, 33(6):668–676, 1990.

[2] Gauri Shah. *Distributed Data Structures for Peer-to-peer Systems*. PhD thesis, Yale University, New Haven, CT, USA, May 2003.

[3] 이 부분의 애매한 점을 지적해 준 김상건 학생에게 감사한다.