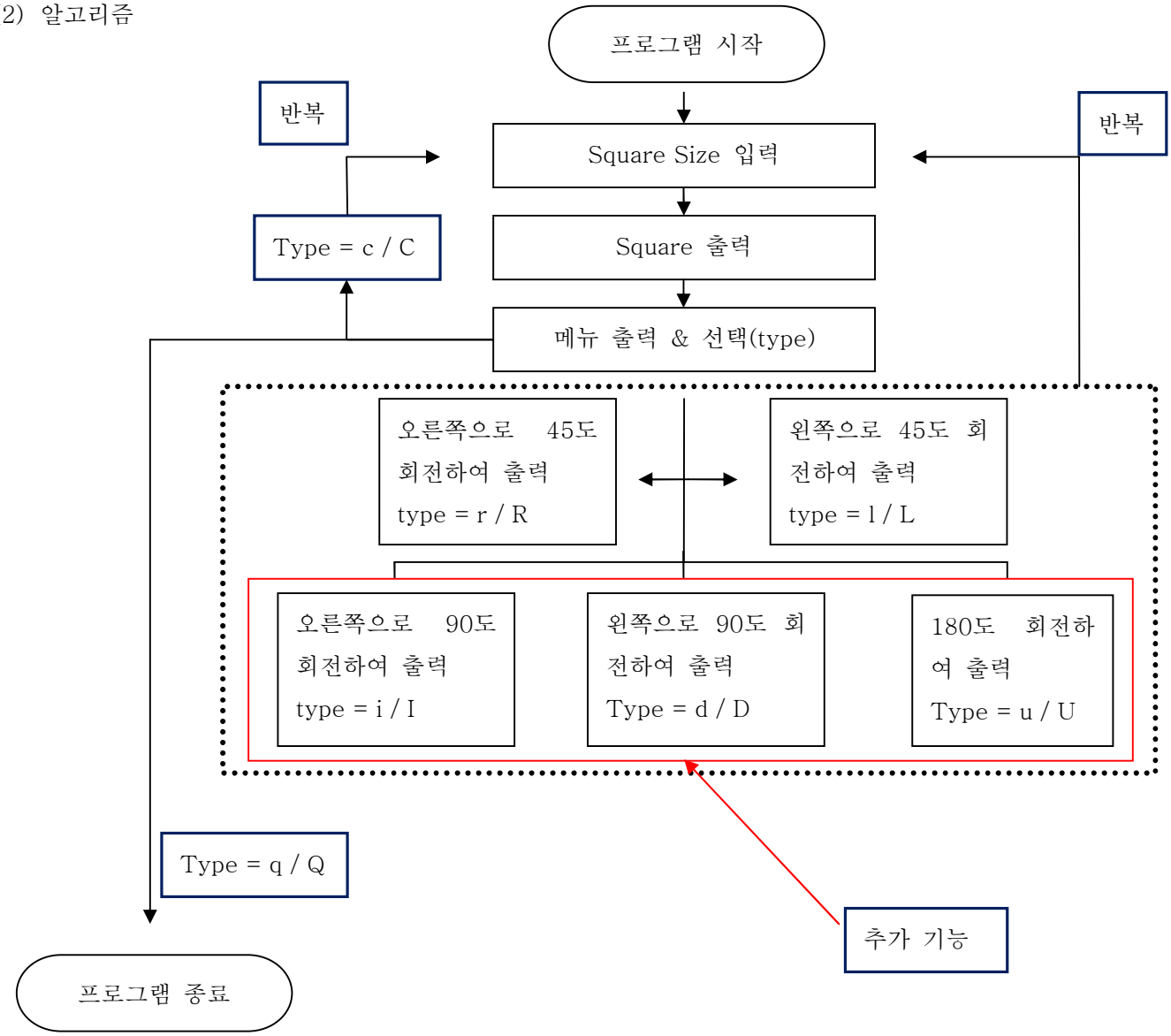
Assign #3

1. Problem #1

(1) 개요

이 프로그램은 정 사면체(nxn)의 숫자들을 여러 가지 회전방향에 따라서 출력하는 프로그램이다. 좌우 45도, 90도 180도 다섯 가지의 회전방법이 있다.

(2) 알고리즘



(3) 전체구조

① Square Size 입력

프로그램을 실행하면 사용자에게 Square Size를 입력할 것을 요청하는 메시지를 출력한다. 사용자가 Square Size를 입력하면 scanf를 통해 int형 변수 size에 저장된다.

이 이후에 입력 값이 integer가 아닌 경우에 scanf()의 return값을 이용하여 이 값이 1이 아닌 경우, 즉 입력이 제대로 되지 않은 경우 다시 입력 받도록 하였다.

② Square 출력

void print_square(int size)

Square Size 입력 받은 뒤에는 그 크기에 맞게 출력한다.

Ex, size=3

1 2 3

4 5 6

7 8 9

③ Rotation type 입력

size를 입력 받은 뒤에는 메뉴를 출력한다.

- 메뉴

Choose rotation
R / r 오른쪽으로 45도 회전
L / l 왼쪽으로 45도 회전
I / I 오른쪽으로 90도 회전
D / d 왼쪽으로 90도 회전
U / u 180도 회전
C / c 새로운 size입력
Q / q 프로그램 종료

Q / q 이외의 메뉴를 선택하면 해당 기능을 수행하고 다시 while size입력 type입력을 반복하게 된다.

④ 회전 부분

void rotate(dint size, int type)

회전은 rotate함수가 담당한다. 이 함수는 size와 type을 인자로 갖고 이 type에 따라서 회전하여 출력한다.

Square 크기를 size, Row의 번호는 i, Column의 번호는 j라고 할 때 각각의 회전에서 위치와 이 번호들 간의 관계는 아래와 같다.

1) 오른쪽으로 45도 회전

45도로 회전할 때는 아래위 반으로 나누어서 출력하였다.

위쪽

j+(size-1)*(j-i)

아래쪽

size*(j-size+1)+(size-1)*(size*2-j-i)

2) 왼쪽으로 45도 회전

45도로 회전할 때는 아래위 반으로 나누어서 출력하였다.

위쪽

size*j-(size+1)*(j-i)

아래쪽

(size*size-j+ size)-(size*2-j-i)*(size+ 1)

3) 오른쪽으로 90도 회전(추가기능)

size*size - (size - j) - i*size

4) 왼쪽으로 90도 회전(추가기능)

(size - j) + size*i

5) 180도 회전(추가기능)

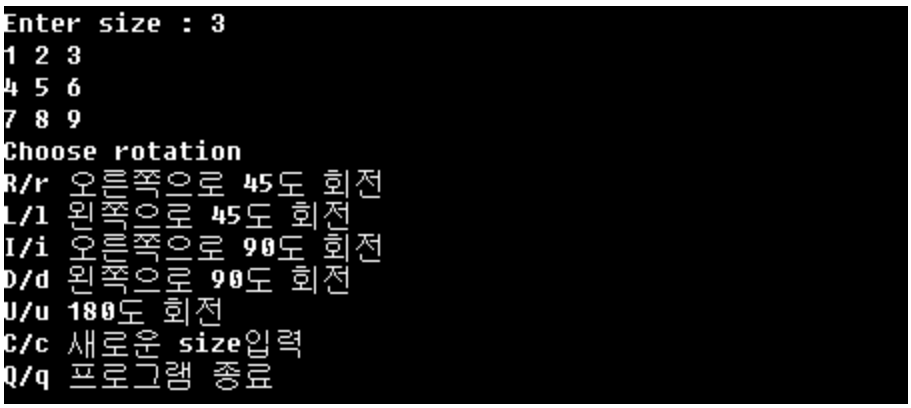
(size)*(size-j) -i

⑤ 프로그램 종료

Type을 입력할 때 q / Q를 입력하면 프로그램이 종료된다.

(4) 프로그램 실행방법 및 예제

프로그램을 실행하면 위와 같은 문구가 뜬다.



Size 를 입력하면 Choose rotation 이란 문구가 뜨고 기능이 설명된다.

r/R 의 실행모습

위의 그림과 같이 원하는 size 를 입력하면 그에 따라 square 가 출력되고 회전을 선택하면 그에 맞게 회전되어 출력되고 다시 size 를 입력 받을 수 있도록 한다.

```
Enter size : 3
1 2 3
4 5 6
7 8 9
Choose rotation
R/r 오른쪽으로 45도 회전
L/l 왼쪽으로 45도 회전
I/i 오른쪽으로 90도 회전
D/d 왼쪽으로 90도 회전
U/u 180도 회전
C/c 새로운 size입력
Q/q 프로그램 종료
q
shell.postech.ac.kr/hiscrepl ~/Assn3 {231}
```

Q 를 입력하면 종료된다.

(추가기능)

```
Enter size : 4
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
```

```
u
16 15 14 13
12 11 10 9
8 7 6 5
4 3 2 1
```

```
i
13 9 5 1
14 10 6 2
15 11 7 3
16 12 8 4
```

```
d
4 8 12 16
3 7 11 15
2 6 10 14
1 5 9 13
```

위의 그림과 같이 u – 180 도 회전, i – 오른쪽으로 90 도 회전, d – 왼쪽으로 90 도 회전 기능을 추가하였다.

(5) 토론

① 2차원 loop

for loop 속에 for loop를 또 만들어서 2차원 loop를 사용하였다. Index가 많이 헷갈렸다.

(6) 결론

첫 번째 문제는 크게 어렵지는 않았다. 각각의 회전에서 row, column, size와 해당 숫자와의 관계를 찾아내는 게 좀 어려웠는데 하나하나 찾아가는 것이 재미있었다.

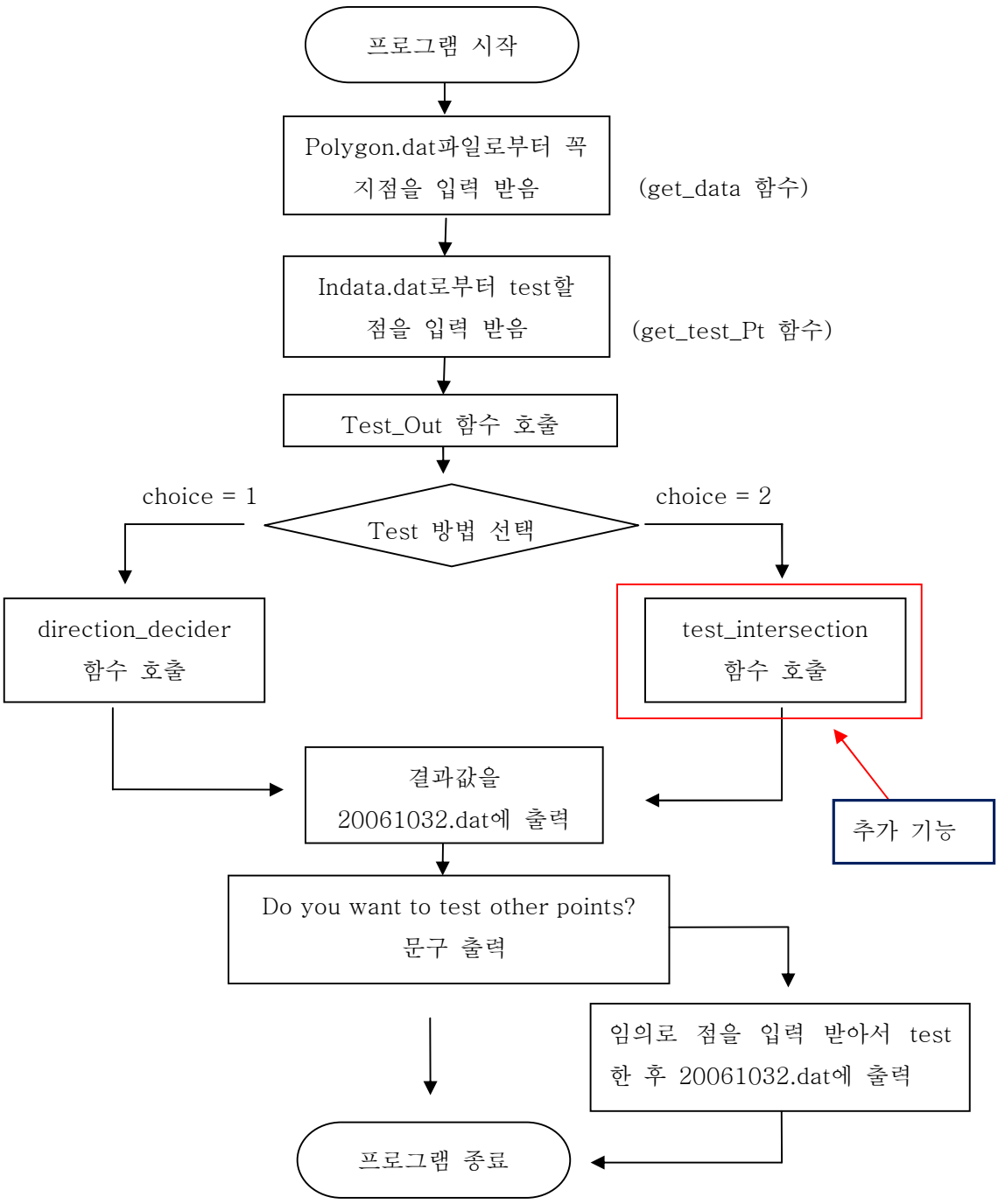
2. Proplem #2

(1) 개요

이 프로그램은 주어진 점으로 이루어진 도형이 있고 입력 받은 점이 이 도형 내부에 있는지 외부에 있는지 판별해서 20061032.dat파일에 그 결과를 inside/outside로 각각 입력하는 프로그램이다. 외부/내부를 판별하는 방법은 두 가지가 있으며 사용자는 이를 선택해서 사용할 수 있다.

도형을 이루는 점은 polygon.dat파일로부터 받고 test할 점은 indata.dat로 부터 받는다. indata.dat의 점들의 test가 끝나면 사용자로부터 임의의 점을 입력 받아서 test해 볼 수 도 있다.

(2) 알고리즘



(3) 전체 구조

① 프로그램 실행

프포그램을 실행하면

//////////////////////////

Inside? Or Outside?

Program Start!

//////////////////////////

문구가 화면에 출력되고 자동으로 get_data함수가 호출된다.

② 꼭지점 입력

```
int get_data(File *in, int Pt[][2])
```

꼭지점 입력은 위의 함수가 담당한다. 파일 포인터 in이 가리키는 파일에 저장되어있는 꼭지점 값을 Pt[][2]에 저장한다. 그리고 이 함수는 입력 받은 꼭지점의 개수를 반환한다.

③ Test할 점 입력

```
int get_test_Pt(FILE *in, int Pt[][2])
```

test할 점은 위의 함수에서 입력 받는다. 파일 포인터 in이 가리키는 파일에 저장되어있는 꼭지점 값을 Pt[][2]에 저장한다. 그리고 이 함수 역시 입력 받은 꼭지점의 개수를 반환한다.

```
while(fscanf(in, "%d %d %d", &i, &Pt[i][X], &Pt[i][Y])!=EOF) i++;
```

코드를 사용하여 파일의 끝까지 값을 입력 받는다.

④ 좌표값 test

```
void Test_Out(FILE* in, int Rslt[], int test[][2], int poly[][2], int test_Pt_num, int poly_num)
```

좌표값이 convexhull내부에 존재하는지 외부에 존재하는지 test는 위의 함수가 담당한다. test하는 방법은 hint1, hint2를 이용하는 방법(direction_decider 함수)과 test할 점의 x좌표를 지나고 x축에 수직인 직선이 convexhull과 이루는 교점의 개수를 이용한 방법(test_intersection 함수)이 있다. 이 두 방법은 모두 점이 외부에 있을 때 0내부에 있을 때 1을 반환한다. 이 값을 바탕으로 Test_Out함수는 inside/outside를 판별하여 20061032.dat파일에 출력한다.

⑤ inside/outside 방법 1

```
int direction_decider(int px, int py, int poly[][2], int poly_num)
```

이 함수는 두 점 (x1,y1), (x2,y2)가 이루는 선분을 기준으로 점(px, py)이 왼쪽에 위치하는가 오른쪽에 위치하는가를 test하는 함수이다.
점(px,py)과 두 점 (x1,y1), (x2,y2)를 잇는 직선간의 관계를 구하는데 아래의 식을 사용한다.

s = (x2-x1)(py-y1) - (px-x1)(y2-y1)

이 계산 결과 s가 음수면 오른쪽 0이면 동일선상 양수면 왼쪽에 존재한다. 이 문제에선 도형 위에 존재하는 경우는 내부에 존재하는 경우로 보았다.

위의 연산을 주어진 꼭지점(문제에선 7개)들이 이루는 선분 7개에 차례로 적용시켜 한 경우라도 오른쪽에 존재하면 외부에 존재하는 점으로 판단하였다.

S가 0보다 크면 0, 작거나 같으면 1을 반환한다.

⑥ inside/outside 방법 2 (추가기능)

int test_intersection(int px, int py, int poly[][2], int num)

이 함수는 주어진 점이 cunvexhull 내부에 있는지 외부에 있는지를 판별하는 또 다른 함수이다. test할 점 (x,y)과, (x,infinity)를 지나는 직선과 cunvexhull간의 교점의 개수로 내부/외부를 판별한다.

한 점이 외부에 존재하면 cunvexhull과의 교점이 짝수
한 점이 내부에 존재하면 cunvexhull과의 교점이 홀수 이다.

(x,y), (x,infinity)를 연결하는 선분과
두점 (x1, y1) (x2, y2)를 연결하는 선분이 만나는지는 아래의 식으로 알 수 있다.

(x1-x)(x2-x)

1) 이때 이 값이 양수면 만나지 않는다.

2) 음수인 경우는 (x1, y1)과 (x2, y2)를 (x1-x) : (x-x2)로 내분하는 점의 y좌표가 y보다 큰지를 확인한다. 크면 만나고 안크면 안만난다.

3) 이 값이 0인 경우는 세 가지 경우를 생각할 수 있다.
   A. x1 = x, x2 = x
      y1, y2가 모두 y보다 크지 않으면 -> 만나지 않음
      y1, y2중 하나만 y보다 크면 -> 주어진 점은 경계선위에 있음
      y1, y2 둘다 y보다 크면 -> x0, x3를 비교하면 됨

   B. x1 = x, x2 != x
      이 경우는 index가 하나 올라가면 C.의 경우와 같아진다.

C. x1 != x, x2 = x
y2가 y보다 크지 않으면 -> 만나지 않음
y2가 y보다 크면 -> x1과 x3를 비교

이 연산을 거쳐 만나는 경우에만 inter 변수에 1을 더하여서 7점이 이루는 7개의 선분의 연산이 끝났을 때 홀수이면 1, 짝수이면 0을 반환한다.

⑦ Auxiliary_test

```
void Auxiliary_test(FILE *out, int Poly_Pt[][2], int poly_num)
```

주어진 점들에 대한 test가 끝나면 사용자로부터 임의로 점을 입력 받아 test를 한다. Scanf를 사용하여 사용자로부터 입력 받을 좌표 수를 입력 받고 그 만큼의 좌표를 입력 받은 후 1번 방법을 사용하여 inside/outside를 판별하여 20061032.dat파일에 출력하였다.

(4) 프로그램 실행방법 및 예제

```
////////////////////
Inside? or Outside?
Program start!!
////////////////////
There are two method to calculate choose 1 or 2:
```

프로그램을 실행하면 위와 같은 문구가 뜬다. 계산하는 방법을 선택할 수 있다.

```
////////////////////
Inside? or Outside?
Program start!!
////////////////////
There are two method to calculate choose 1 or 2: 1
Points in file tested!
Do you want to test other points[Y/N]? N
shell.postech.ac.kr/hiscrepl ~/Assn3 {233}
```

1선택, 하나를 선택하면 test가 되고 임의의 점을 test할 것인지 묻는다.

```
////////////////////
Inside? or Outside?
Program start!!
////////////////////
There are two method to calculate choose 1 or 2: 2
Points in file tested!
Do you want to test other points[Y/N]?
```

2선택, 결과는 같다.

```
0 outside
1 outside
2 outside
3 inside
4 outside
5 inside
6 inside
7 outside
8 outside
9 outside
10 outside
11 outside
12 outside
13 outside
14 outside
15 outside
16 outside
17 outside
18 outside
19 outside
```

위 그림과 같이 결과가 나온다(20번 이하 생략)

(추가기능)

```
////////////////////
Inside? or Outside?
Program start!!
////////////////////
There are two method to calculate choose 1 or 2: 1
Points in file tested!
Do you want to test other points[Y/N]? Y
Input the number of points you want to test(1~100): 3
Input your point ex) (3,2) => 3 2
1st point: 50 70
2nd point: 30 20
3rd point: 10 5
Done!
```

위와 같이 임의의 점을 입력하면 아래의 그림처럼 100개의 test 결과 아래 결과가 출력된다.

```
99 outside

Bellows are auxsiliary points
(50, 70) is inside
(30, 20) is outside
(10, 5) is outside
```

(5) 토론

① Inside/outside를 판별하는 방법 비교

이 프로그램에선 Inside/outside를 판별하는데 두 가지 방법을 사용한다.

첫 번째 방법: (x2-x1)(py-y1) - (px-x1)(y2-y1)

두 번째 방법: (x1-x)(x2-x)

첫 번째 두 번째 모두 위와 같이 비슷한 연산을 먼저 수행한다. 그런데 첫 번째 방법은 연산결과 값이 양수냐 아니냐 만을 가지고 따지고 하나라도 양수라면 외부의 점으로 판단할 수 있다.

그런데 두 번째 방법은 양수, 음수, 0으로 나누고 0인 경우에도 다른 조건을 더 만족시켜야 한다.(test_intersection 함수 설명 참조) 따라서 두 번째 방법이 더 나은 방법이라고는 할 수 없겠다.

② Array 활용

수업시간에 array를 배웠다. Array의 이름은 포인터처럼 사용할 수 있었다. 그런데 이번에 새롭게 알게 된 것은 함수를 선언할 때에 2차원 배열의 경우에는 낮은 차원의 element가 몇 개인지 써주어야 한다는 것이다.
예를 들어 int mat[20][5];라고 선언했다고 한다면 함수를 선언할 때에는 mat[][5]라고 5를 꼭 붙여줘야 하는 것이었다.

③ 파일 입출력

이번 어싸인에서는 입력과 출력을 파일로 했다. c프로그램을 통해서 문서 파일을 만들 수 있는 것이 놀라웠다.

(6) 결론

이번 어싸인은 좀 어려웠다. Convexhull의 inside/outside를 판단하는 방법을 생각하는 것이 쉽지 않았다. 결국 주어진 힌트보다 좋은 방법은 찾을 수 없었으나 다른 방법은 찾을 수 있었다. 또 한가지 아쉬웠던 점은 배열을 선언 할 때 처음부터 개수가 주어지지 않고 사용자가 숫자를 입력하면 그 개수에 맞게 배열을 만들 수 있게 만들고 싶었으나 그렇게 할 수 없었다.

줄이자면 조금 어렵긴 했지만 재미있는 어싸인 이었다.

3. make file

Makefile은 아래와 같이 구성하였다.

```
p1: Assn3_1.o
	gcc -o p1.exe  Assn3_1.o
p2: Assn3_2.o
	gcc -o p2.exe  Assn3_2.o

Assn3_1.o: Assn3_1.c
	gcc -c Assn3_1.c
Assn3_2.o: Assn3_2.c
	gcc -c Assn3_2.c
```