Object-Oriented Programming Report

Assignment 1-1

Professor	Donggyu Sim
Department	Computer engineering
Student ID	2020202031
Name	Jaehyun Kim

Class (Design / Laboratory)	1 / B
Submission Date	2023. 3. 17

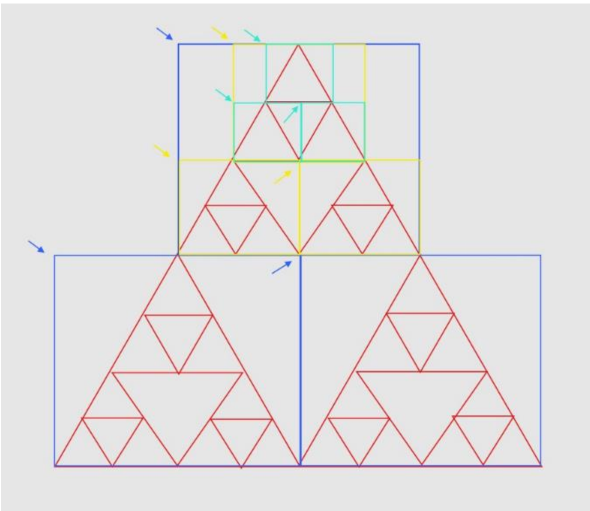
Program 1

❑ 문제 설명

문제를 자신이 이해한대로 설명, 프로그램 작성시에 고려할 모든 개념 서술, 구현 방법 설명

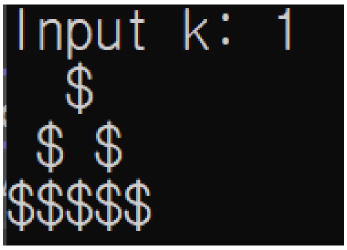
정수 k 를 입력 받고 N 에 2^{k-1}을 저장합니다. 여기서 N 은 가장 큰 삼각형의 한 변에 가장 작은 단위의 삼각형이 몇 개 나열 돼있는가를 의미하는 것으로 해석했습니다. 따라서 k 가 최대값 8 일 때, N 은 128 이 되고, 가장 큰 삼각형의 밑변의 길이는

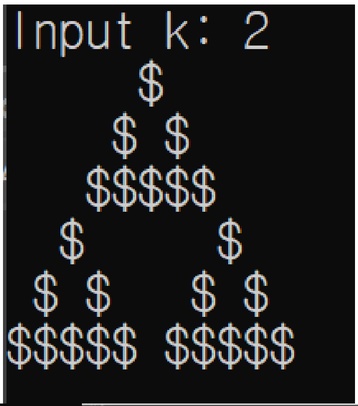
6N-1=767, 높이는 3N=384 이므로 384x767 이차원배열을 전역변수로 선언했습니다.

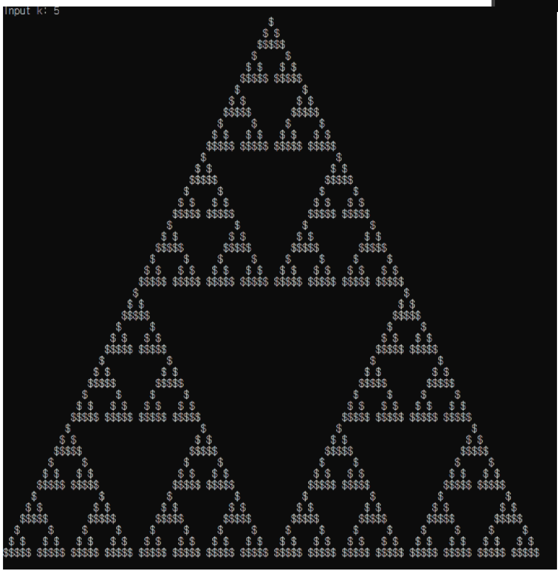


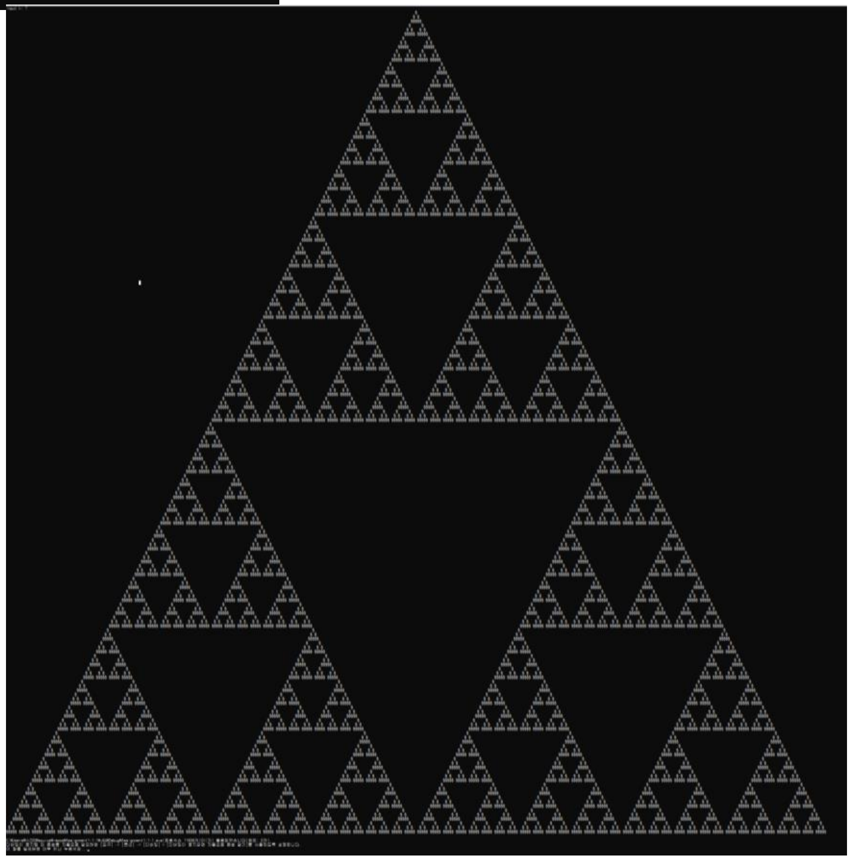
makeTri 함수를 보겠습니다.시에르핀스키 삼각형은 가장 작은 단위의 삼각형이 나올 때까지 계속해서 세 개의 삼각형으로 나눠 들어갈 수 있습니다. 이러한 반복형태에서 재귀적 영감이 떠올랐습니다. makeTri 가 호출될 때, 화살표가 가리키는 곳의 좌표를 매개변수로 전달했습니다.

❑ 결과 화면









$N = 2^{k-1}$이므로 k 가 1 증가할 때, N 은 2 배가 되는 것을 알 수 있다. 출력 결과에서도 확인할 수 있듯이, k 가 1 증가하면 N 이 2 배가 되면서 큰 삼각형의 한 변에 포함된 삼각형의 개수 또한 2 배가 된다는 것을 알 수 있다.

❑ 고찰

변수 k 를 unsigned char 형으로 선언해서 cin 으로 입력 받은 수를 k 에 저장하고 N=pow(2, k-1)을 실행했더니 N 에 터무니없이 큰 값이 저장됐습니다. k 가 unsigned char 로 선언됐기에, cin 으로 입력받을 때, k 에 문자 숫자가 저장됩니다(ex. 1 입력 -> '1'저장). 따라서 정수형 숫자와 문자 숫자 간의 차인 48 을 k 에서 뺐더니 문제가 해결됐습니다.

Program 2

❑ 문제 설명

근의 공식을 통해 이차방정식의 해를 구하는 문제입니다. 먼저 x^2, x의 계수와 상수를 float 형 변수 a, b, c 에 입력 받습니다. 근의 공식$x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$을 통해 이차방정식의 근을 구해 출력해줍니다.

유사한 두 실수끼리의 뺄셈을 하면 round-off error 가 발생하기 때문에

$x = \frac{-b+\sqrt{b^2-4ac}}{2a}$의 경우 분모 분자에 각각 $b + \sqrt{b^2 - 4ac}$를 곱해줌으로써

$x = \frac{-4ac}{2a(b+\sqrt{b^2-4ac})}$로 식을 변형시켜, 뺄셈 연산을 덧셈연산으로 바꿔줬습니다.

❑ 결과 화면

```
a: 0
b: 1
c: 1
Unexpected factor of a quadratic term
```

$0x^2 + bx + c = 0$은 $bx + c = 0$과 같기 때문에 일차방정식이므로 a 의 입력이 잘못됐다.

```
a: 1
b: -2
c: 1
X1, X2: 1 (double root)
```

판별식 $D = b^2 - 4ac = 0$이므로 중근을 가진다.

```
a: 1
b: 1
c: 1
The equation has no real number solutions.
```

판별식 $D = b^2 - 4ac < 0$이므로 실근을 가지지 않는다.

```
a: 1
b: 62.1
c: 1
X1: -0.0161072, X2: -62.0839
```

판별식 $D = b^2 - 4ac > 0$이므로 두 실근을 가진다.

❑ 고찰

컴퓨터가 실수를 저장할 때 근사값을 저장하는 방식 때문에 유사한 값을 가지는 두 실수끼리의 뺄셈이 round-off error 를 증폭시킨다는 것을 이해하는 것은 큰 문제가 없었습니다. 하지만 근의 공식을 어떻게 변형시켜야 하는지에 대한 아이디어를 떠올리던 중, 유리화가 떠올랐습니다. 중학교에서는 분모의 유리화를 배우는데 역으로 분자를 유리화 함으로써 유사한 두 실수 간의 뺄셈을 덧셈으로 바꿀 수 있었습니다. a=1, b=62.1, c=1 의 예시에서도 X1=-0.0161072 가 잘 출력되는 것을 확인했습니다.

Program 3

❑ 문제 설명

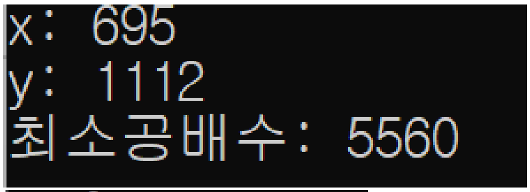
재귀적 정의로 최대공약수를 구하는 함수 GCD 를 구현하고 GCD 를 이용하여 최소공배수를 구하는 함수 LCM 을 구현하는 문제입니다.

GCD 를 구현할 때, 유클리드 호제법을 참고했습니다.

또한 수를 입력 받을 때 큰 수, 작은 수가 순서 상관없이 입력될 수 있기 때문에, 항상 x 가 y 보다 크도록 swap 함수를 정의해 줬으며, 최소공배수는 두 수로 나눠떨어지는 가장 작은 양수라고 정의됐기에, 음수를 입력 받을 때를 대비하여 음수를 양수로 바꿔주는 makePositive 함수도 정의했습니다.

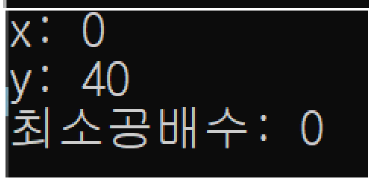
(x, y 의 최소공배수) = (x * y / 최대공약수) 라는 점을 참고하여 LCM 을 구현했습니다.

❑ 결과 화면

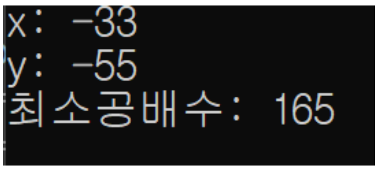


695 와 1112 의 최대공약수는 139 이므로

최소공배수는 $695 \times 1112 \div 139 = 5560$이다.



0 으로 나누어 떨어지는 수는 0 자신밖에 없다.



최소공배수는 양수이기 때문에 음수 또한 양수처럼 계산한다.

❑ 고찰

```
x: 2147483647
y: 2147483646
최소공배수: -2147483646
```

int 자료형의 최대값과 최대값-1 의 최소공배수를 계산해봤더니 음수가 출력됐습니다. 이 출력결과가 의미하는 것은 오버플로우임이 자명했습니다. 그래서 int 자료형의 최대값과 최대값-1 의 최소공배수를 직접 구할 수는 없었기에, 가능한 가장 큰 최소공배수는 두 수의 곱이라고 판단하여 두 수의 곱을 연산한 결과 4,611,686,011,984,936,962 였습니다. 이 수를 포함하는 범위를 가진 자료형으로 long long 을 선택하여 getLCM 의 반환형을 long long 으로 하고 매개변수로 받은 x, y 를 long long 으로 강제 형변환하여 계산을 진행했습니다.

```
x: 2147483647
y: 2147483646
최소공배수: 4611686011984936962
```

다음과 같이 오버플로우 없이 연산결과가 잘 출력됨을 확인할 수 있었습니다.

Program 4

❑ 문제 설명

3x3 행렬을 int 형 3x3 배열로 입력받습니다. 그 후 입력받은 행렬로 det A, C, C^T를 차례로 구해줍니다. $A^{-1} = \frac{1}{\det A} C^T$공식을 이용하여 역행렬을 구하는 프로그램을 작성하는 문제입니다.

역행렬을 구하는데 필요한 행렬의 개념은 이공계 대학수학 교재를 참고했습니다.

A=$\begin{matrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{matrix}$ 일 때,

행렬식 det A = $a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{23}a_{31}) + a_{13}(a_{21}a_{32} - a_{22}a_{31})$

여인수행렬 C 의 요소 c_{ij} = $(-1)^{i+j}M_{ij}$

※M_{ij}는 A 의 i 행, j 열을 제외시킨 행렬의 행렬식이다.

C 의 전치행렬 C^T는 C 의 열과 행을 뒤집은 행렬이다.

C-> C^T에서는 C=$\begin{matrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{matrix}$ 빨간펜으로 표시한 부분만 참조하여 행과 열을 뒤집어주면 C^T를 구할 수 있다.

❑ 결과 화면

```
2 2 1
-1 1 0
0 0 0
The inverse matrix does not exist.
```

$A^{-1} = \frac{1}{\det A} C^T$에서 det A = 0 이 되면 역행렬이 정의가 되지 않는다. 위의 주어진 행렬의 행렬식 det A = 0 이므로 "The inverse matrix does not exist"라는 문구가 출력됩니다.

```
1 2 3
0 1 4
5 6 0
-24     18      5
20      -15     -4
-5      4       1
```

C^T의 요소들이 det A 로 나눠떨어져서 역행렬이 정수로 문제없이 잘 출력됩니다.

```
1 0 2
2 -1 0
1 1 1
-0.2  0.4   0.4
-0.4  -0.2  0.8
0.6   -0.2  -0.2
```

C^T의 요소들이 det A 로 나눠떨어지지 않아 역행렬이 소수로 출력됐습니다. 하지만 수치상으로 정답이 맞습니다.

❑ 고찰

3x3 행렬을 int 형 3x3 배열에 입력 받고, 역행렬을 저장하기 위한 3x3 int 형 배열을 선언했습니다. 그리고 det A, C, C^T를 차례로 구했습니다. 그러나 $A^{-1} = \frac{1}{\det A} C^T$공식을 적용하여 역행렬을 구하는 부분에서 문제가 발생했습니다. C^T의 요소들이 크면 문제가 없었지만 크기가 작아서 det A 로 나눠떨어지지 않을 경우에 int 형이 소수를 나타내지 못해 0 으로 출력되는 문제가 있었습니다. 이에 역행렬 3x3 배열을 double 형으로 선언했더니 문제가 해결됐습니다.