2021년 1학기

**프로그래밍과 문제해결**

Assignment #1

담당 교수 : 윤은영

학번 : 20210084

학과 : 무은재학부

이름 : 김지민

POVIS ID : kjm1672

================명예서약(Honor code)===============

“나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.”

==============================================

1. 문제의 개요

본 프로그램을 간략히 소개하면 다음과 같다.

* 4개의 메뉴를 설정하여 사용자로부터 메뉴의 번호를 입력 받는다.
* 각 메뉴에 해당하는 기능을 수행한다.

1번 : 리만 합

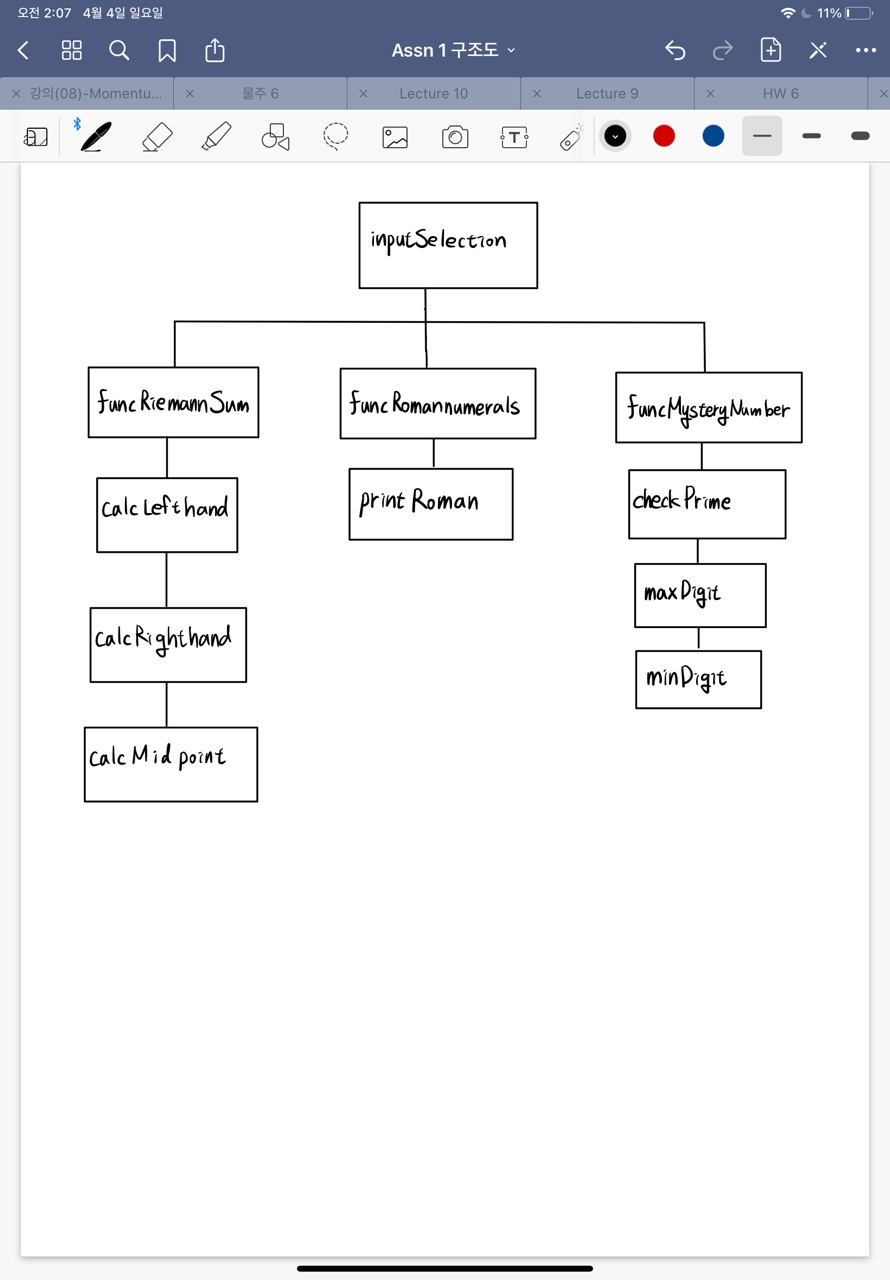
2번 : 로마 숫자 변환

3번 : 미스터리 숫자 생성

4번 : 프로그램 종료

* 1번의 시행 이후 반복 수행 여부를 묻고 입력 값 (y or n)에 따라 결정한다.
* 리만 합은 사용자로부터 2차 함수의 각 항의 계수를 입력 받아 왼쪽, 오른쪽, 가운데 리만 합을 모두 출력해야 한다.

이때 구조 차트 (structure chart)는 아래와 같이 표현될 수 있다.

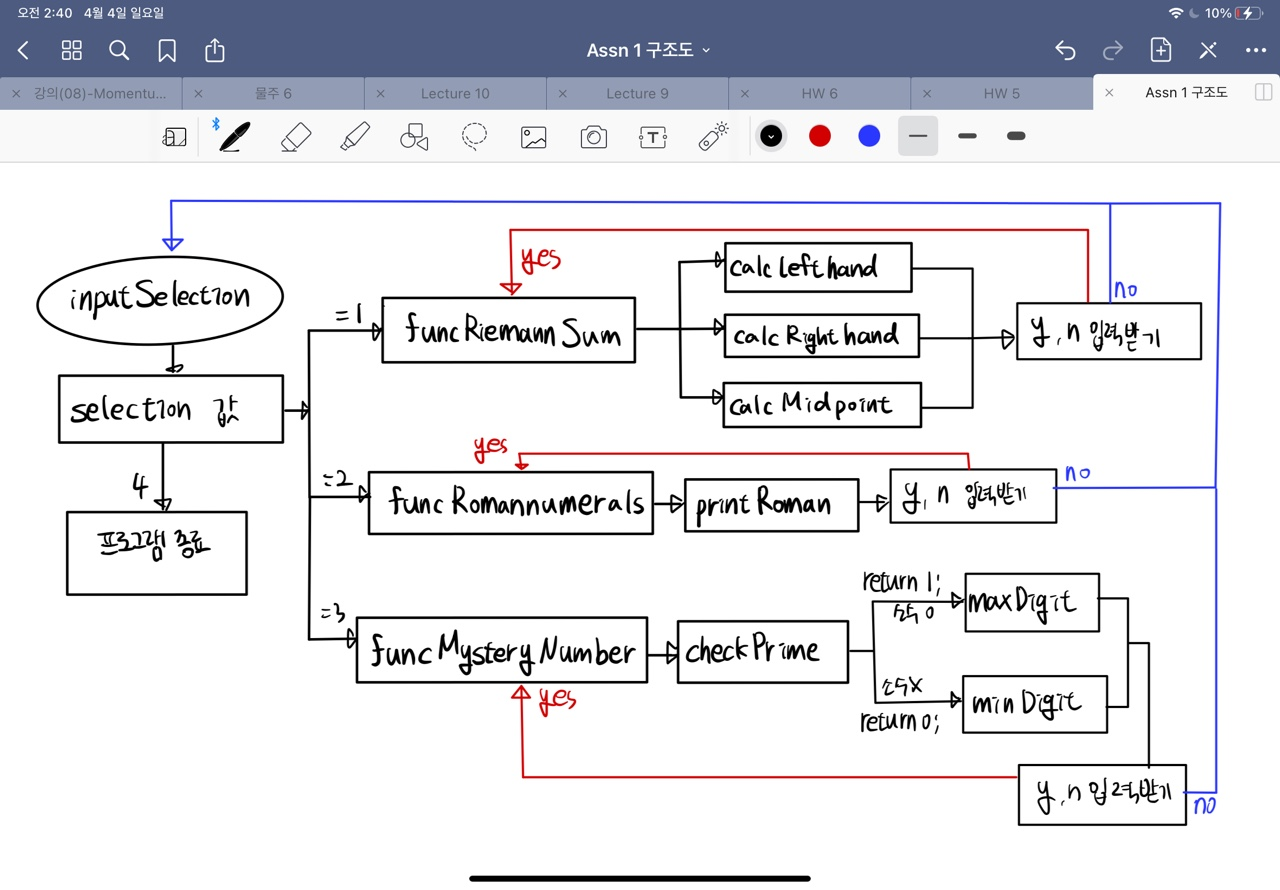


1. 알고리즘

본 프로그램 작성을 위한 알고리즘을 Presudo 코드 형태로 나타내면 다음과 같다.

|  |
| --- |
| Presudo-algorithm |
| 1. 헤더파일 <stdio.h>, <math.h>를 포함시킵니다. 2. 메뉴 화면을 출력하고 사용자로부터 메뉴 선택을 받습니다.   -이때 1~4 범위에서 벗어난 정수가 입력되면 다시 입력 받습니다.  3. 선택된 메뉴의 기능을 수행합니다.  -1번: funcRiemannSum 함수 호출, 리만 합을 출력. 왼쪽, 오른쪽, 가운데 리만 합 모두를 출력합니다.  (a) : 사용자로부터 이차함수의 각 항의 계수를 입력 받습니다.  (b) : 넓이를 구하고 싶은 x의 범위를 입력 받습니다.  (c) : 근사할 사각형의 개수를 입력 받습니다.  (d) : calcLefthand, calcMidpoint, calcRighthand 함수를 호출하여 각 합을 구하고 출력합니다.  사각형의 넓이 = 사각형 1개의 폭\*함수값. (이때 함수값의 위치를 사각형의 어디로 하냐에 따라 왼쪽, 오른쪽, 가운데 리만 합이 결정됩니다.)  -2번: funcRomannumerals 함수 호출, 입력된 정수를 로마 숫자로 변환.  (a) : 사용자로부터 1~1000 내 정수를 입력 받습니다.  - 범위 밖 숫자가 입력될 시, 다시 입력 받음.  (b) : printRoman 함수를 호출하여 입력값을 로마 숫자로 변환. -> switch 문을 이용  -3번 : funcMysteryNumber 함수 호출, 입력값을 미스터리 숫자로 변환.  (a) : 사용자로부터 1~1000000000 내 정수를 입력 받습니다.  - 범위 밖 숫자가 입력될 시, 다시 입력 받음.  (b) : 각 자릿수를 분리한 후, n번째 숫자를 n제곱한 후 다 더합니다. 이 수를 sum으로 지정합니다. -> for문을 이용합니다.  (c) : checkPrime 함수를 호출하여 sum이 소수인지 판별합니다. 소수이면 원래 입력값의 가장 큰 자릿수를 더하고, 아니면 가장 작은 자릿수를 뺍니다. 이때, minDigit, maxDigit 함수를 호출하여 가장 작은 자릿수와 큰 자릿수를 판별합니다.  (d) funcMysteryNumber 함수로 돌아와 출력합니다.  -4번: “Program Terminated.”를 출력하고 프로그램을 종료합니다.  4. 4번 메뉴를 제외한 모든 메뉴의 함수는 한번 시행 후 반복 시행의 여부를 사용자로부터 입력 받아야 합니다. ‘n’을 입력 받았을 시, 다시 처음 화면으로 돌아가야 합니다. |

위 알고리즘을 flowchart 로 표현하면 아래와 같다.



1. 프로그램 구조 및 설명
2. 초기 화면 출력

* int inputSelection() 함수 : 각 번호에 해당하는 기능이 적힌 기본 화면을 출력한다. 사용자로부터 입력값을 받고, (단, 정수 외 문자는 고려하지 않는다.) main 함수로 입력값을 반환한다.
* main 함수 : inputSelection()을 호출하고 받은 입력값을 판단한다. 입력값이 1, 2, 3일 경우에는 해당 함수를 호출하고, 4일 경우 "Program Terminated."을 출력하고 프로그램을 종료한다. 이때 while문과 if문을 구현함으로써, 해당 함수의 기능이 끝나고 사용자로부터 ‘n’를 입력 받았을 때 다시 처음부터 수행이 가능하도록 하였다.

1. Riemann Sum

* void funcRiemannSum() : 사용자로부터 2차항의 계수를 각각 입력 받는다. 변수 ini, fin에 x의 범위를 입력 받고 변수 d에 근사할 사각형의 개수를 입력 받는다. 이후 각각의 리만 합을 구하는 함수들을 호출하여 입력 받은 값들을 이용하여 3개의 리만 합을 도출하고 출력한다. 각 합들은 %.4f를 이용하여 소수 넷째 자리까지 출력한다.
* void calcLefthand : 왼쪽 리만 합을 계산하고 출력한다. 코드에 공식을 다음과 같이 구현할 수 있다.

sum += interval\*(a \* pow(ini + interval \* i, 2) + b \* (ini + interval \* i) + c);

for문을 이용하여 사각형의 넓이를 누적한다. 이때 i의 범위는 0부터 (사각형 개수 -1)까지다.

* void calcRighthand : 오른쪽 리만 합을 계산하고 출력한다. 코드에 공식을 다음과 같이 구현할 수 있다.

sum += interval\*(a \* pow(ini + interval \* i, 2) + b \* (ini + interval \* i) + c);

i의 범위는 1부터 사각형 개수까지다.

* void calcMidpoint : 오른쪽 리만 합을 계산하고 출력한다. 코드에 공식을 다음과 같이 구현할 수 있다.

sum += interval\*(a \* pow(ini + interval \* i + interval/2, 2) + b \* (ini + interval \* i+ interval / 2) + c); i의 범위는 0부터 (사각형 개수-1)까지다.

1. 로마 숫자 변환

* void funcRomannumerals() : 사용자로부터 정수를 입력 받는다. 이때 1~1000이라는 범위를 벗어날 경우 while문을 이용하여 경고 메시지를 출력하고 다시 입력 받는다. (음수일 경우 : "The number should be positive" / 1000 초과일 경우 : "The number should be epual to or less than 1000”) printRoman 함수를 호출하여 로마 숫자를 출력한다. 2)와 마찬가지로 반복 수행 여부를 묻는다. 이때 함수 전체 부분에 while문을 걸어서 입력값이 y이면 반복하고, n이면 while문 밖으로 나가 함수를 끝내고 다시 main 함수로 돌아가 처음 화면이 출력된다.
* void printRoman : int ones, tens, hund에 입력값을 일의 자리, 십의 자리, 백의 자리로 구분하여 저장한다. 만약 값이 1000이면 그에 해당하는 로마 숫자 “M”을 출력하도록 따로 설정한다. 각 자리 숫자에 해당하는 로마 숫자를 switch문을 이용하여 출력한다. 이때 백의 자리, 십의 자리, 일의 자리 순으로 출력되어야 하므로 switch문 순서 또한 그에 맞춘다.

1. 미스터리 숫자 생성

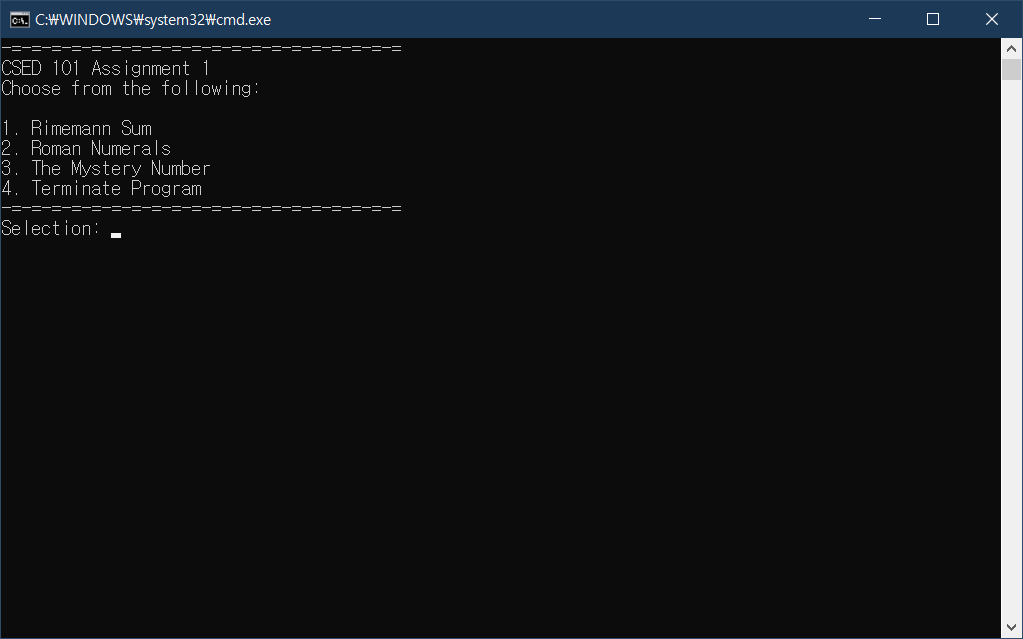
* void funcMysteryNumber : int num, num2, num3를 선언하여 사용자로부터 받은 입력값을 세 변수에 저장한다. While문과 if문을 이용하여 값을 1~1,000,000,000 내로 받을 수 있도록 한다. for문을 이용하여 변수 count에 몇 자릿수인지 저장한다. 이때 num을 계속 10으로 나눠 0이 될 때 for문을 멈추는 방식으로 구할 수 있다. n번째 자릿수를 n제곱한 뒤 더한 값인 sum 역시 같은 방식을 이용한다. num변수는 count를 구하는 과정에서 0이 되었으므로 num2를 이용한다. Sum 공식은 다음과 같은 나타낼 수 있다. 이때 i는 0부터 시작한다.

sum += pow(num2 % 10, (count - i));

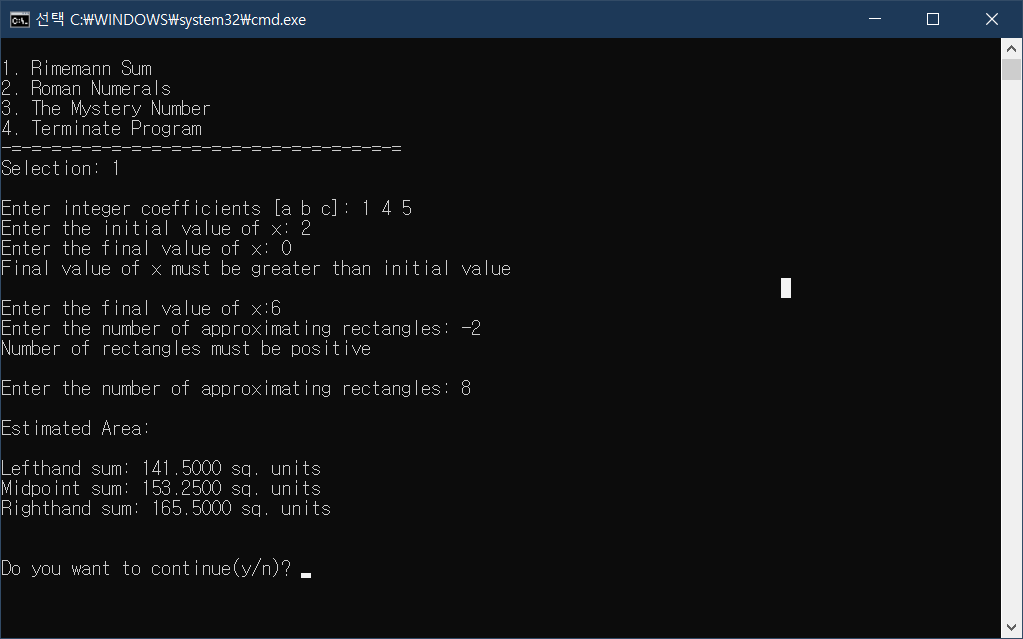
이후 checkPrime 함수를 호출하여 sum 값이 소수인지 아닌지 판별한다. maxDigit, minDigit 함수에 원래 숫자를 보내 가장 큰 자릿수와 작은 자릿수를 구한다. 이때num2은 sum을 구하는 과정에서 0이 되었으므로 num3을 이용한다. checkPrime 함수가 보낸 값이 1이면 (sum+가장 큰 자릿수)를 출력, 0이면 (sum-가장 작은 자릿수)를 출력한다.

* int checkPrime : 변수 sum 값을 받아 소수인지 아닌지 판별한다. 1과 sum 값을 제외한, sum보다 작은 모든 수로 나눴을 때 나머지가 0인 경우가 1번이라도 있으면 소수가 아니니 0을 반환, 없으면 소수이니 1을 반환한다. for문과 if문을 이용하여, 나머지가 0인 경우가 생기면 break로 반복문을 나오도록 구현한다. 또한 10, 100과 같이 sum이 1인 경우 따로 if문으로 설정하여 0을 반환하도록 한다. (1은 소수이므로)
* int maxDigit : num3의 값을 받아 각 자릿수를 비교한다. 변수 max에 일의 자릿수를 초기값으로 설정한다. 그 다음 if문과 삼항 연산자를 이용하여 초기값보다 다음 자릿수가 크면 max의 값을 큰 수로 바꾸는 과정을 반복하고 최종 max값을 반환한다.
* int minDigit : num3의 값을 받아 각 자릿수를 비교한다. 변수 min에 일의 자릿수를 초기값으로 설정한다. 그 다음 if문과 삼항 연산자를 이용하여 초기값보다 다음 자릿수가 작으면 min의 값을 작은 수로 바꾸는 과정을 반복하고 최종 min값을 반환한다.

1. 프로그램 실행방법 및 예제

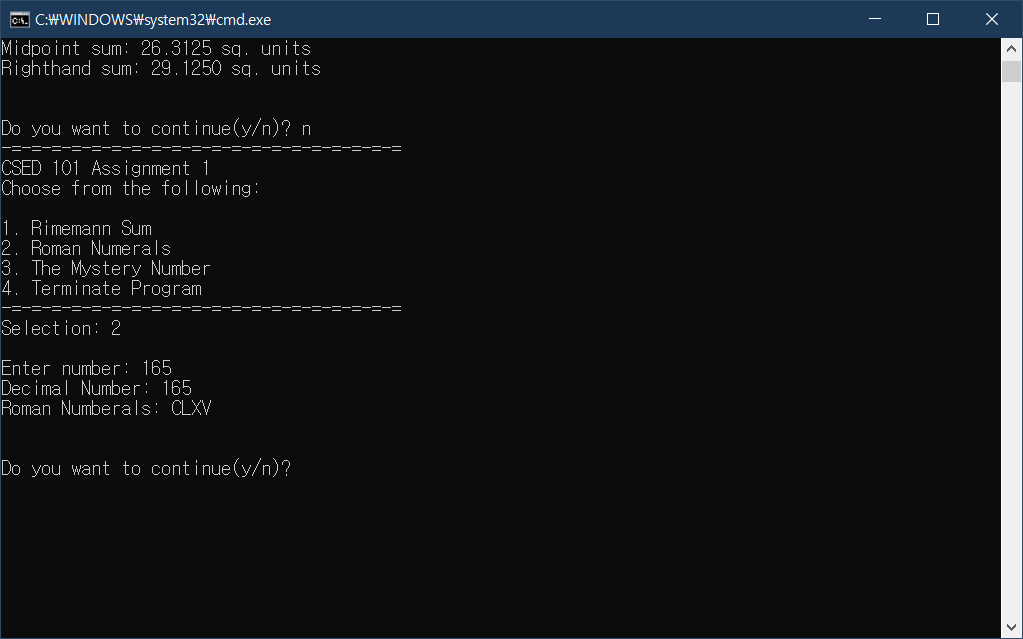


<초기 메뉴 선택 화면>

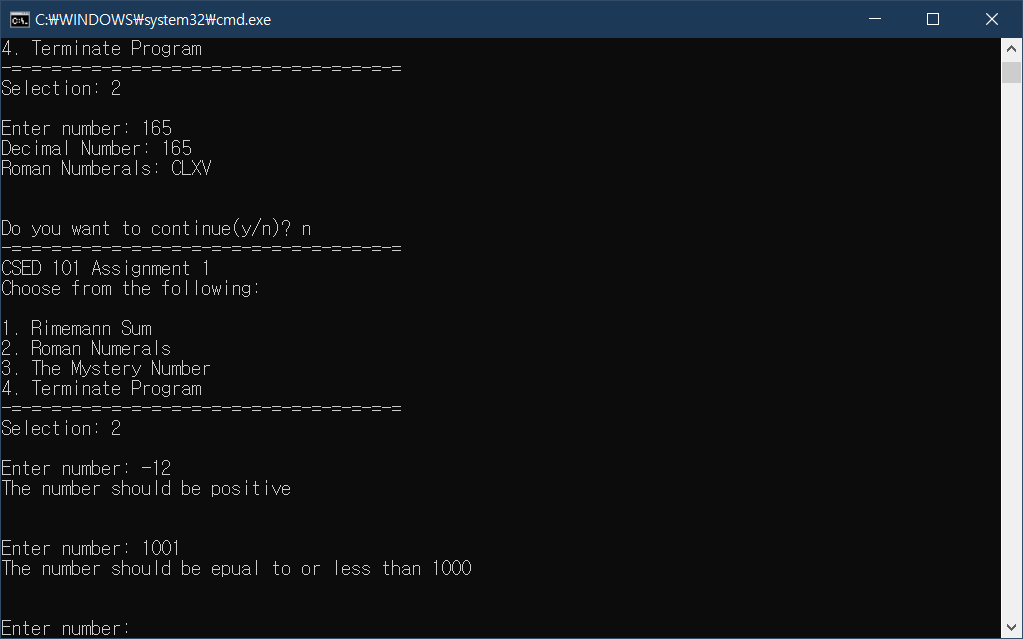


<1을 입력했을 때>

* X의 마지막 값이 처음 값보다 작아서 경고 메시지가 출력된 것을 확인할 수 있다.
* 사각형 개수에 음수를 입력하면 경고 메시지가 출력되는 것을 확인할 수 있다.
* 계속 할 것인지 묻는 메시지가 나오는 것을 확인할 수 있다.



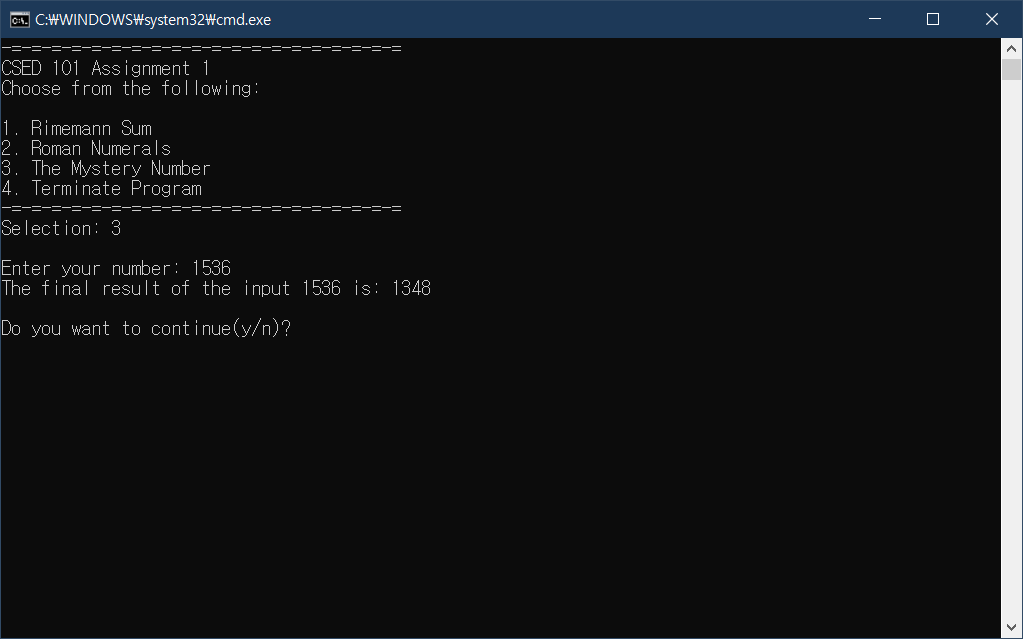
<2를 입력했을 때>



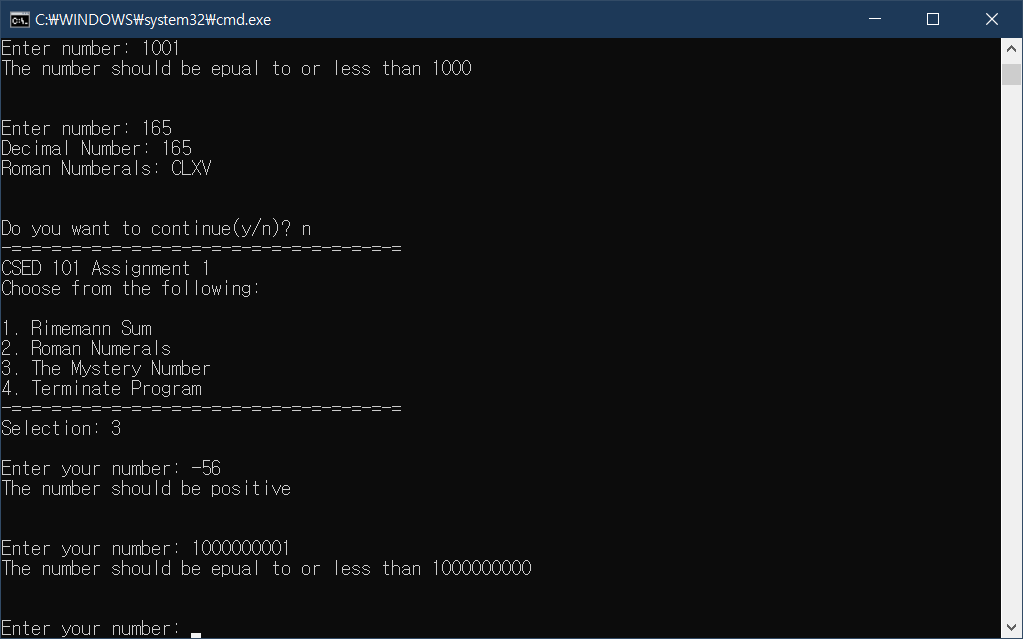
<2를 입력했을 때 경고 메시지>

* 음수를 넣었을 때
* 1000보다 큰 수를 넣었을 때

경고 메시지를 확인 할 수 있다.

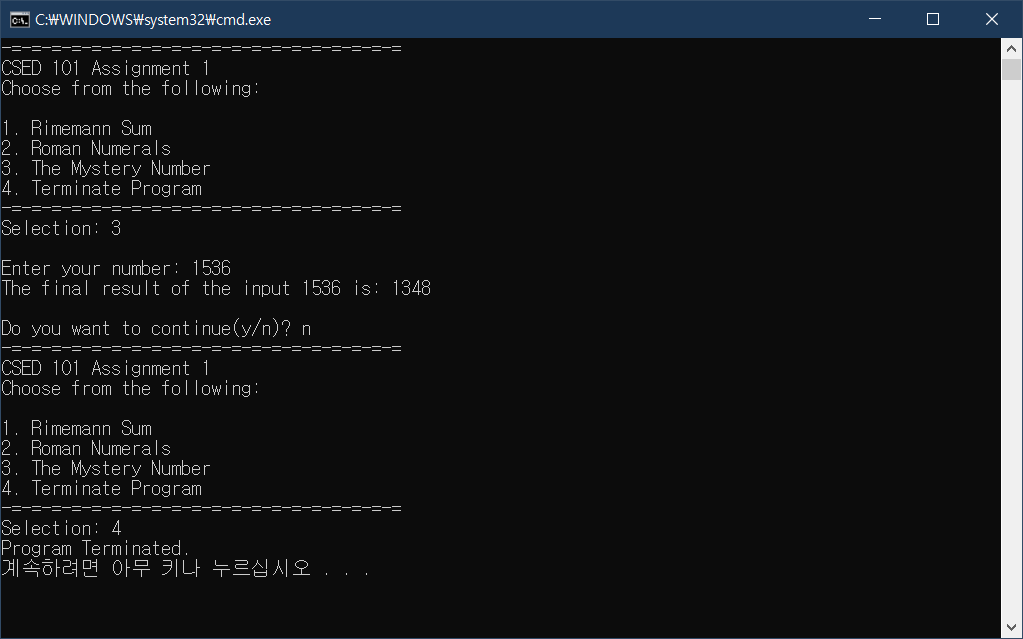


<3을 입력했을 때>

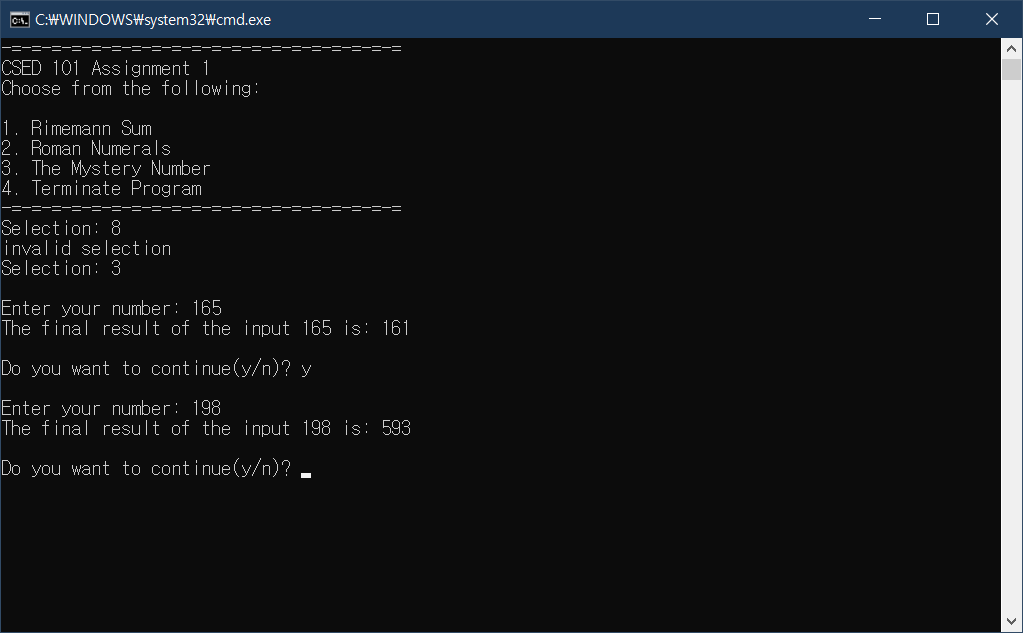
<3을 입력했을 때 경고 메시지>

* 음수를 입력했을 때
* 1,000,000,000보다 큰 수를 입력했을 때

경고 메시지가 출력되는 것을 확인할 수 있다.

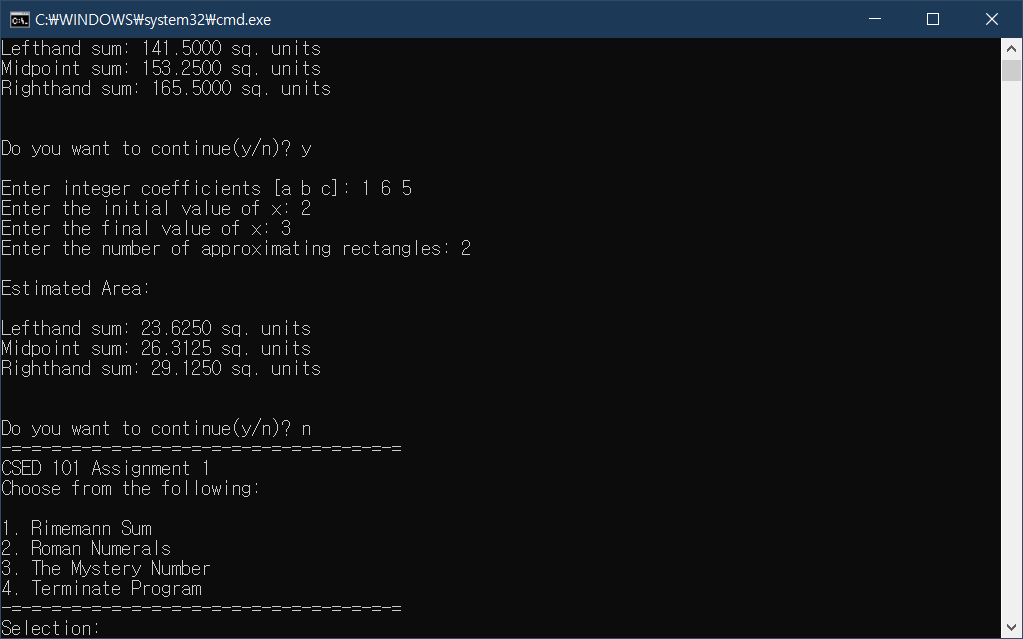
<4를 입력했을 때>

* 프로그램이 종료되는 것을 확인할 수 있다.



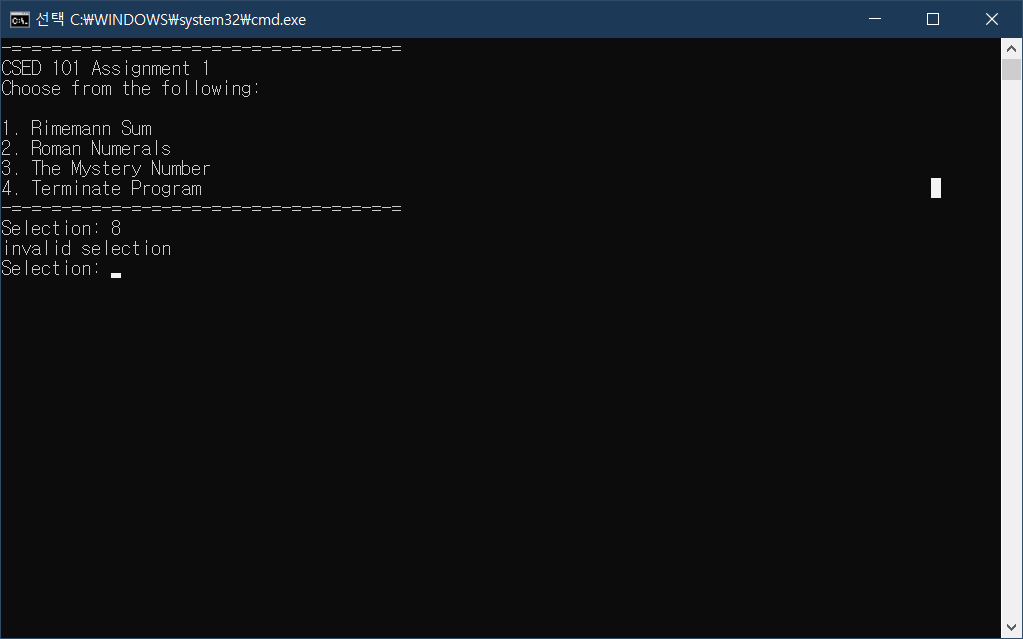
<y를 입력했을 때>

* 같은 함수가 반복 수행되는 것을 확인할 수 있다.



<n을 입력했을 때>

* 다시 처음 메뉴 화면으로 돌아오는 것을 확인할 수 있다.



<1, 2, 3, 4 외 숫자를 입력했을 때>

* 경고 메시지가 출력되면서 다시 입력 받는 것을 확인할 수 있다.

1. 토론

* 함수의 결과값이 출력되고 y or n를 입력 받아야 하는 상황에서 scanf 함수가 제대로 입력 받지 않는 문제가 발생하였다.
* 이는 엔터가 입력 버퍼에 입력되었기 때문이다.
* 이를 해결하기 위해 printf와 scanf사이에 임의의 변수에 getchar()함수를 호출하여 입력하면 입력 버퍼가 비워지면서 문제가 해결된다.

1. 결론

* 본 과제는 조건문과 반복문을 사용하는 방식을 스스로 고민하고 알고리즘을 구성하는 데에 유용했다. Presudo 코드와 구조도를 그리는 과정을 통해 많은 기능이 있는 프로그램을 어떻게 코딩할 것인지 고민하는 시간을 가질 수 있었고 좀 더 체계적으로 진행할 수 있었다.

1. 개선방향

* 본 과제에서 요구하는 사항들은 지금까지 배운 수업 내용을 적절히 활용하면 충분히 해결할 수 있었다. 다만 본인이 Presudo 코드 작성이나 알고리즘 구상에 미숙하기 때문에 프로그램의 구조의 윤곽이 잡히는 데에 시간이 많이 소요되었다. 이번 경험을 발판 삼아 앞으로 연습을 꾸준히 한다면 더욱 향상될 것이라 생각한다.