객체지향프로그래밍설계 과제 보고서

2021202085 전한아솔

1. Write a program that reads 5 integers and prints the minimum, maximum, and average values. (If a decimal part occurs, it must be rounded.)

```
Enter the five numbers: 10 6 29 31 88
MIN: 6
MAX: 88
AVG: 33
```

5개의 수를 입력받아서 최댓값, 최솟값, 평균을 출력하는 프로그램으로 평균은 반올림해서 출력해야 한다.
크기가 5인 int형 배열을 선언하여 수를 입력받고 이중 for문을 사용하여 앞 인덱스의 숫자가 뒤 인덱스보다 크다면 자리를 바꾸는 방식을 사용하여 배열을 오름차순으로 정렬했다. 따라서 [0]에는 최솟값이, [4]에는 최댓값이 저장된다.
배열에 입력된 값을 double형 변수에 모두 더하여 5로 나눈 후, 양수라면 0.5를 더해주고 음수라면 0.5를 빼서 int형으로 형 변환하여 반환해서 반올림을 구현했다.

```
Enter the five numbers: 10 6 29 31 88
MIN: 6
MAX: 88
AVG: 33
```

(int)를 사용하여 강제로 형 변환을 진행하면 소수점 뒷자리가 없어지기 때문에 0.5를 더하거나 빼서 반올림을 구현할 수 있었다.

2. Write a program that print a 2-dimension array (5 × 10) with numbers randomly generated and also print sum and average of each row with '|' as a separator. The random numbers should be integer from 1 to 99. The computed average value should be presented with an integer number. (If a decimal part occurs, it must be rounded.) (The spaces between numbers must be as follow: one space or two spaces or three spaces)

```
 4 91 51 99 81  6 88 22 95 28 | 565 | 57
56 77  3 62  7 32 85 48 24 63 | 457 | 46
95  2 83 45 12  4 58 47 35 12 | 393 | 39
31 26 17  7 60 16 24 83 79 27 | 370 | 37
 1  8  3  6  8  9  1  1  9  7 |  53 |  5
```

1부터 99의 랜덤한 값을 (5×10) 크기의 2차원 배열에 저장하여 각 행의 합과 평균을 출력해주는 프로그램이다. 여기서도 마찬가지로 평균은 반올림을 해주어야 하고 랜덤한 숫자가 한

자릿수거나 합이 두 자릿수라면 오른쪽으로 정렬해야 한다.
srand의 시드로 time(NULL)을 넣어서 매 실행마다 랜덤한 값이 바뀌도록 했으며 1부터 99까지의 숫자가 랜덤으로 할당되어야 하므로 rand()%99+1로 값을 할당하여 1부터 99까지의 숫자만 랜덤으로 할당되게 했다. 각 행과 열을 2중 반복문을 통해 돌면서 sum 변수에 계속 더해준다. 따라서 sum에는 각 행의 합 값이 저장된다. 각 행에서 마지막의 값이 한 자릿수라면 오른쪽으로 정렬해야 하므로 공백을 먼저 출력하고 숫자를 출력해준다. 또한 각 행의 마지막 값을 출력한 후에는 더 이상 랜덤 값을 출력하지 않아도 되므로 break를 사용하여 반복을 탈출한다. 마지막 값이 아닐 때, 아까와 같은 방법으로 한 자릿수라면 오른쪽으로 정렬해야 하므로 공백을 먼저 출력해주고 숫자를 출력하고 아니라면 바로 숫자를 출력한다.
sum에 저장된 합값 또한 두 자릿수라면 오른쪽으로 정렬을 해주어야 하므로 if문을 통해서 10이상, 100이하의 값이라면 공백을 출력하여 오른쪽으로 정렬하고 아니라면 공백은 제외하고 숫자만 출력한다.
평균값은 1부터 99의 수만 할당되므로 양수이다. 따라서 0.5를 더하고 int형으로 강제 형변환을 통해 반올림을 구현하여 반올림된 평균값을 출력한다.

```
71  39  78  85  73  67  14  31  36   6 |  500 |  50
44  27  23  30  88  19  19  74  43  97 |  464 |  46
85  86  79  10  29  87  94  79  97  99 |  745 |  75
69  39  98  27  45  64  83  61  88  92 |  666 |  67
57  58  32  59  80   7  77   7  14   5 |  396 |  40
```

랜덤하게 할당된 숫자 중 한 자릿수의 숫자는 공백을 하나 추가하여 오른쪽으로 정렬됨을 볼 수 있고, 평균값 또한 반올림되어 출력되는 것을 볼 수 있다.

3. Write a program that reads 10 characters and prints the first and the last on one line, the second and the ninth on the next line, the third and seventh on the next line, and so forth. Sample input and the results are shown below.

```
Enter ten characters: a b c d e E D C B A
Input characters are:
a A
b B
c C
d D
e E
```

char형 변수를 10개 입력받아 맨 앞과 맨 끝, 2번째와 9번째, 3번째와 8번째... 의 순서로 값을 출력해주는 프로그램이다.
for문을 사용하여 i=0부터 9까지 반복시킨다. 그리고 [i]와 [9-i] 인덱스를 출력하면 (0,9), (1,8), (2,7)...의 인덱스가 출력되므로 문제의 조건을 만족시킨다.

```
Enter ten characters: a b c d e E D C B A
Input characters are:
a A
b B
c C
d D
e E
```

4. Write a function that takes a positive integer value and returns the number with its digits reversed. For example, given the number 8263, the function should return 3628.

```
Enter the number: 8263
Reversed number: 3628
```

```
Enter the number: 2000000001
Reversed number: 1000000002
```

```
Enter the number: 2630
Reversed number: 362
```

입력받은 양의 정수를 거꾸로 출력해주는 프로그램이다. 하지만 입력하는 숫자가 0으로 끝나면 0은 빼고 출력한다.

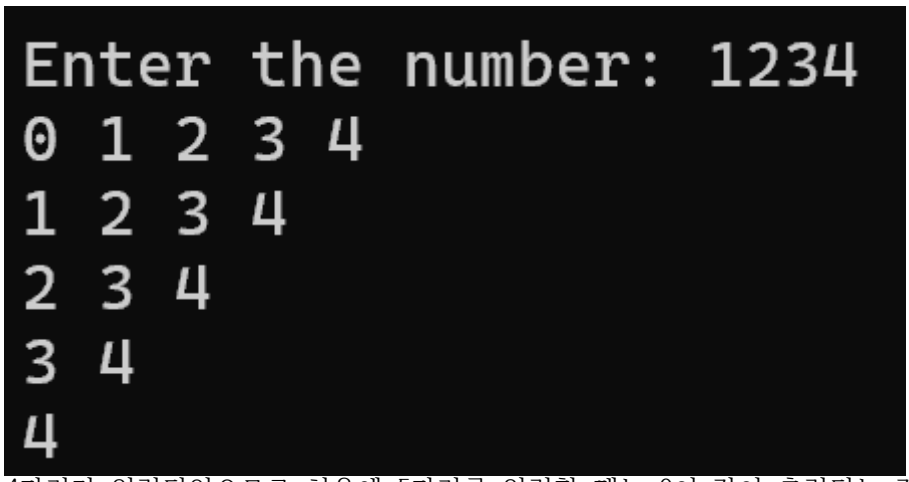
입력받은 양의 정수를 한 자리씩 나누어서 배열에 저장한다. 이때 양의 정수이므로 100자리를 넘어가지 않기에 배열의 크기는 100으로 생성했다. 또한 입력받는 수를 의 자릿수를 세어주는 cnt라는 변수를 0으로 초기화하여 생성했다. 입력받는 수가 0이 아니게 될 때까지 num%10을 사용하여 1의 자릿수를 뽑아내어 [cnt] 인덱스에 저장하고 num/10을 통해서 1의 자릿수를 없애 주고 cnt를 1증가시킨다. 이렇게 0이 될 때까지 반복하면 입력받은 수를 한 자리씩 배열에 저장할 수 있다. 0부터 cnt까지 반복문을 배열에 저장된 값을 새로운 int형 변수에 더하고 10을 곱해준다. 하지만 마지막 반복에서는 더해주기만 하고 10은 곱하지 않아도 되므로 if문을 통해서 cnt-1이 아닐 때만 10을 곱하도록 구현했다. 배열에는 입력받은 정수가 한 자리씩 거꾸로 저장되어 있으므로 새로운 변수는 입력받은 정수가 거꾸로 저장된다.

```
Enter the number: 123456789
Reversed number: 987654321
```

5. Write a program that prompts a user for an integer value in the range 0 to 32,767 and then prints the individual digits of the numbers on a line with two spaces between the digits. The first line is to start with the leftmost digit and print all five digits; the second line is to start with the second digit from the left and print four digits, and so forth. For example, if the user enters 1234, your program should print.

```
Enter the number: 1234
0  1  2  3  4
1  2  3  4
2  3  4
3  4
4
```

0부터 32,767까지의 입력을 받는다. 5자리 이하라면 0을 넣어주고 왼쪽으로 한 칸씩 밀어서 5개, 4개 ,3개... 의 숫자를 출력해주는 프로그램이다.
크기가 5인 배열을 0으로 초기화하여 선언했다. 4번 문제에서 사용한 방법과 같은 방법으로 한 자리씩 % 연산자를 사용하여 저장했는데, 입력받은 수와 같은 순서로 저장하기 위해서 4번째 인덱스부터 1씩 줄여가며 저장했다. 행과 열을 출력하기 위해서 i와 j를 사용하여 2중 반복문을 사용했다. I가 하나씩 증가하며 행을 바꾸므로 j는 I부터 4까지의 인덱스를 출력하도록 구현하여 왼쪽으로 하나씩 밀려서 출력되도록 구현하였다.



4자리가 입력되었으므로 처음에 5자리를 입력할 때는 0이 같이 출력되는 것을 볼 수 있고, 줄이 바뀔 때마다 왼쪽으로 하나씩 밀려서 출력되는 결과를 볼 수 있다.

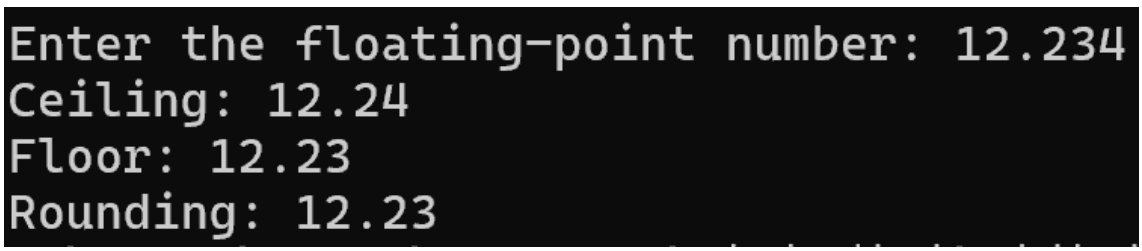
6. Write a program that reads a floating-point number and prints the ceiling, floor, and rounding values without using standard library functions in C and C++. This program calculates to two decimal places. (If the third decimal place is less than 5, the absolute value is rounded down. If the third decimal place is 5 or more than 5, the absolute value is rounded up.)

```
Enter the floating-point number: 12.234
Ceiling: 12.24
Floor: 12.23
Rounding: 12.23
```

```
Enter the floating-point number: 1.035
Ceiling: 1.04
Floor: 1.03
Rounding: 1.04
```

```
Enter the floating-point number: -1.035
Ceiling: -1.03
Floor: -1.04
Rounding: -1.04
```

입력받는 정수를 올림, 내림, 반올림하여 소수 두 번째 자리까지 출력해주는 프로그램이다. 이 프로그램에서 반올림은 사사오입을 적용한다.
먼저, 올림은 양수라면 1을 더하고 강제 형 변환을 통해서 소수점을 버림으로써 구현할 수 있고, 음수라면 강제 형 변환을 해주면 음수의 올림이 구현된다.
두 번째로 내림은 양수라면 강제 형 변환을 통해서 소수점을 버리고, 음수라면 1을 빼고 형 변환을 해주어서 음수의 내림을 구현한다.
마지막으로 반올림은 양수라면 0.499를 더하여 반환하고 음수라면 0.499를 빼고 반환하여 반올림이 구현되도록 했다.
입력받은 수가 num이라면 각 함수에서 100을 곱하여 소수 세 번째 자리부터 확인하도록 했고 다시 100을 나누어서 소수 세 번째 자리에서 올림, 내림, 반올림이 구현되도록 하였다.
소수점 아래 자릿수 고정을 위해서 fixed를 사용했고, precision에 시드로 2를 전달하여 소수점 아래 두 자릿수까지만 출력하도록 했다.

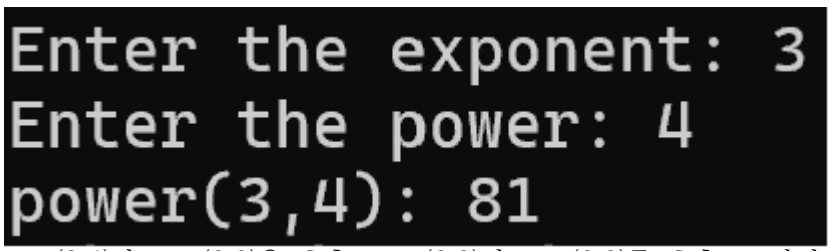


소수 세 번째 자리가 4이므로 반올림을 적용하면 12.23이 나오는 것을 볼 수 있다.

7. Write a recursive function power(base, exponent)that, when invoked, returns $base^{exponent}$. For example, $\text{power}(3,4) = 3 \times 3 \times 3 \times 3$. Assume that exponent is an integer greater than or equal to 0. Hint: The recursion step would use the relationship: $base^{exponent} = base \times base^{exponent-1}$, And the termination condition occurs when exponent is equal to 1 because $base^0 = 1$.

```
Enter the base: 3
Enter the exponent: 4
power(3,4): 81
```

base와 exponent를 입력받아 base^exponent을 계산하는 프로그램으로, 재귀함수를 사용하여 함수를 정의해야한다.
base^n=base^(n-1)이라는 힌트와 base^0은 1이라는 힌트가 주어졌으므로 이를 이용하면, exponent가 0이라면 1을 반환하고 0이 아니라면 base*base^(exponent-1)을 반환한다. 이렇게 되면 exponent값이 1씩 줄어들면서 다시 power 함수를 호출하므로 power은 재귀함수이다. 이렇게 exponent값이 줄어들다가 0이 되면 1이 반환되면 호출된 함수가 역순으로 호출되면서 base^exponent의 값이 반환된다.



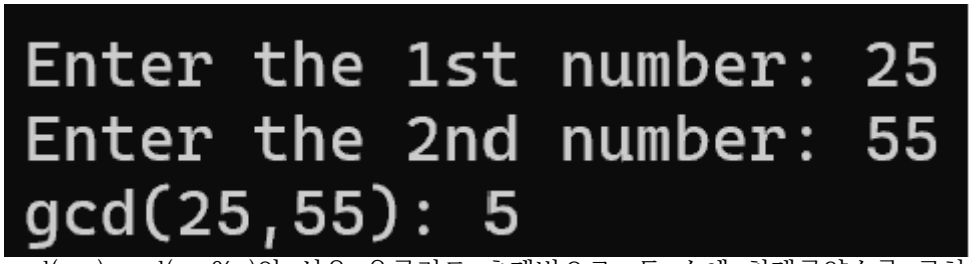
pow(3,4)가 pow(3,3)을 호출, pow(3,3)이 pow(3,2)를 호출.... 마지막에 pow(3,0)이 호출되면 1이 반환되고 호출된 함수가 역순으로 호출되므로 pow(3,1), pow(3,2)... pow(3,4)가 계산되어 3^4인 81이 반환된다.

8. The greatest common divisor of integers x and y is the largest integer that evenly divides both x and y. Write a recursive function gcd that returns the greatest common divisor of x and y, which is defined recursively as follows:
   If y is equal to 0, then $gcd(x,y)$ is x; otherwise, $gcd(x,y)$ is $gcd(y, x \,\%\, y)$, where % is the modulus operator.

```
Enter the 1st number: 12
Enter the 2nd number: 18
gcd(12,18): 6
```

x와 y를 입력받아서 최대공약수를 구해주는 프로그램이다.
gcd가 최대공약수를 구해주는 함수라고 할 때 gcd(x,y)=gcd(y,x%y)라는 힌트가 주어졌고
y가 0이라면 gcd(x,y)=x라는 조건도 주어졌으므로 gcd함수는 y가 0이라면 x를 반환하고 아니

라면 gcd(y,x%y)를 반환한다. 다시 gcd 함수에 파라미터로 y와 x%y를 전달하며 함수의 두 번째 인자가 0이 될 때까지 반복한다. 함수의 두 번째 인자가 0이 되면 첫 번째 인자를 반환하며 함수가 호출된 역순으로 함수를 실행한다. 이는 재귀함수의 방식이다.

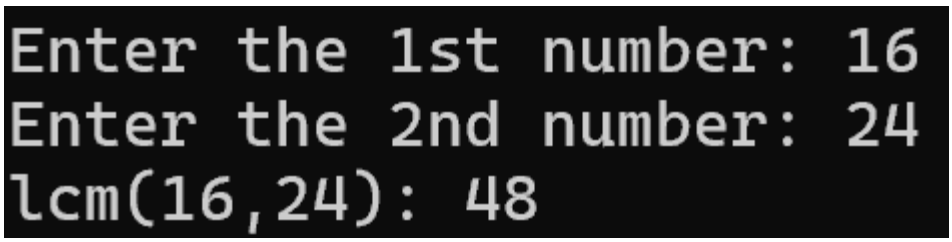


gcd(x,y)=gcd(y,x%y)의 식은 유클리드 호제법으로, 두 수에 최대공약수를 구하는 알고리즘에서 사용하는 방법임을 이번 문제를 통해서 알게 되었다.

9. The least common multiple of integer x and y is the smallest positive integer that is a multiple of both x and y. Write a function $lcm(x, y)$ that returns the least common multiple of x and y using $gcd(x, y)$ function of the problem 8.

```
Enter the 1st number: 12
Enter the 2nd number: 18
lcm(12,18): 36
```

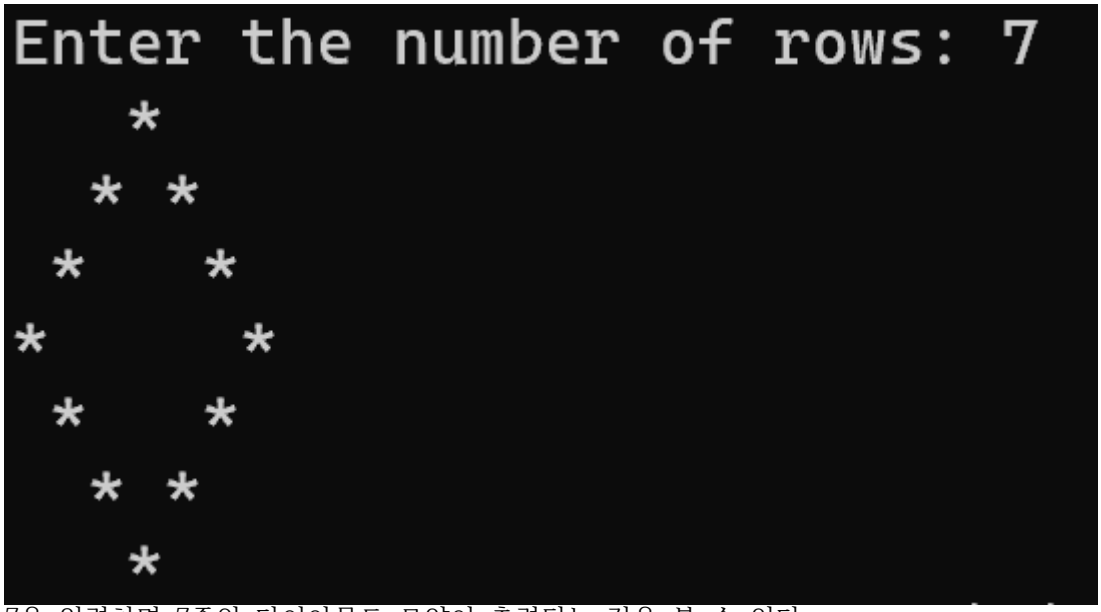
8번 문제에서 구현했던 gcd함수를 통해서 최소공배수를 구하는 lcm함수를 구현하는 프로그램이다.
입력받은 두 수를 x, y라고 한다면 x*y=gcd(x,y)*lcm(x,y)이다. 따라서 최소공배수는 두 수를 곱한 값에서 최대공약수를 나누어주면 구할 수 있다.




```
Enter the number of rows: 3
 *
* *
 *
```

```
Enter the number of rows: 5
  *
 * *
*   *
 * *
  *
```

1부터 19까지의 홀수를 입력받으면 그에 맞는 다이아몬드 모양의 별이 출력되도록 하는 프로그램이다.
행을 i라고 하고 열을 j, 입력받는 수를 num이라고 하자.
처음부터 중간 행(temp=(num+1)/2)까지는 j가 temp를 기준으로 앞뒤 i만큼은 별을 출력하고 나머지는 공백을 출력하도록 했다. temp 행부터 마지막 행까지는 j가 temp를 기준으로 앞뒤 num-i-1만큼은 별을 출력하고 나머지는 공백을 출력하도록 했다. 중간 행을 기준으로 위·아래의 별을 출력하는 열이 다르므로 행을 나누어서 구현하였다.



7을 입력하면 7줄의 다이아몬드 모양이 출력되는 것을 볼 수 있다.

11. Write a function named *multiple* that takes a pair of integers as arguments. This function should determine if the second integer is a multiple of the first integer, returning true if so, and false otherwise.

```
Enter the 1st number: 3
Enter the 2nd number: 12
multiple(3,12): true
```

```
Enter the 1st number: 5
Enter the 2nd number: 12
multiple(5,12): false
```

두 개의 정수를 입력받아 두 번째 정수가 첫 번째 정수의 배수라면 true, 아니라면 false를 출력하는 프로그램이다.
int형 변수 두 개를 인자로 가지는 함수를 정의하여 두 번째 정수를 첫 번째 정수로 나눈 나머지가 0이라면 배수이므로 true를 출력, 0이 아니라면 배수가 아니므로 false를 출력하도록 구현했다.

```
Enter the 1st number: 4
Enter the 2nd number: 12
multiple(4,12): true
Enter the 1st number: 5
Enter the 2nd number: 17
multiple(5,17): false
```

12. The Fibonacci series $0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots$ starts with the numbers 0 and 1, where each subsequent number is the sum of the two preceding numbers.

    (a) Write a non-recursive function $Fibonacci\_iter(n)$ that calculates the n-th Fibonacci number and return it.

    (b) Write a recursive function $Fibonacci\_rec(n)$ that calculates the n-th Fibonacci number and return it.

```
Enter the number: 9
Fibonacci_iter(9): 21
Fibonacci_rec(9): 21
```

첫째 항이 0, 둘째 항이 1이고 앞선 두 수를 더한 수를 나열해놓은 수열을 피보나치 수열이라고 한다. a는 비재귀 함수를 통해서 피보나치 수열의 n번째 값을 반환하는 함수를 구현하고 b는 재귀함수를 통해서 피보나치 수열의 n번째 값을 반환하는 함수를 구현해야한다.
먼저 재귀함수로 구현한 피보나치 수열을 보자.
피보나치 수열의 시작은 0, 1이므로 함수의 인자인 num이 1이라면 0, 2라면 1을 반환한다. 3 이상의 값이라면 num-1을 인자로 전달한 함수와 num-2를 인자로 전달한 함수를 더한 값을 반환한다. 이 때 자기 자신과 같은 함수가 호출되고 인자의 값이 1이나 2가 될 때까지 호출하여 값을 반환 후 호출된 함수가 역순으로 값을 반환한다. 따라서 이 함수는 재귀함수이다.
비재귀 함수로 구현한 피보나치 수열에서는 배열을 사용했다.
int형 포인터 arr을 선언하고 인자로 전달된 num 크기의 int형 사이즈를 동적으로 배열에 할당했다. arr[1]=0, arr[2]=1로 저장한 후, for문을 통하여 3부터 num까지 반복하며
arr[i]=arr[i-1]+arr[i-2]을 통하여 앞선 두 배열의 값을 더하여 저장했다. 이렇게 구현하면 배열에 i번째 인덱스에는 피보나치 수열의 i번 째 값이 저장된다.
위의 방법은 동적 프로그래밍(Dynamic programming)으로 재귀함수는 자신과 같은 함수를 여러번 호출해야하기 때문에 많은 메모리와 시간을 소모하는데, 동적 프로그래밍 기법은 한 번의 계산을 통해서 피보나치 수열을 구현하기 때문에 재귀함수보다 효율적이다.

```
Enter the number:9
Fibonacci_iter(9): 21
Fibonacci_iter(9): 21
```

13. Write a program that converts the lowercase (or uppercase) letter to the uppercase (or lowercase) without using standard library functions in C and C++. (The input string can consist of up to 100 characters.)

Enter the string: 2024 KWangWoON uNiveRSiTY. Result: 2024 kwANGwOon UnIVErsIty.

알파벳 대문자를 소문자로, 소문자를 대문자로 바꾸어주는 프로그램이다.
string을 입력받아야 하므로 100크기의 char형 배열을 선언해주었다. 또한 공백까지 입력을 받아야 하므로 getline을 사용하여 공백까지 char형 배열 str에 저장했다. 배열을 함수의 인자로 전달받아 for문을 사용하여 0부터 배열의 길이-1까지 반복하여 모든 인덱스를 대·소문자 변환하도록 구현했다. 아스키코드에서 알파벳 대·소문자의 차이는 32이다. str[i]가 대문자라면 소문자보다 32만큼 작으므로 32를 더해주면 소문자로 변환이 가능하다. 소문자는 대문자보다 32만큼 크므로 32를 빼주어야 대문자로 변환이 가능하다.

```
Enter the string: 2024 KWangWoON uNiveRSiTY.
Result: 2024 kwANGwOon UnIVErsIty.
```

알파벳 대문자와 소문자의 아스키코드 값이 차이가 나는 것을 알고 있었으나, 이번 과제를 통해서 대·소문자는 32만큼 차이가 난다는 것을 알았고 대문자 A는 65, 소문자 a는 97의 값을 가진다는 것도 배웠다.

14. An integer is said to be a perfect number if the sum of its factors, including 1 (but not the number itself), is equal to the number. For example, 6 is a perfect number, because $6 = 1 + 2 + 3$.

    (a) Write a function $Perfect(n)$ that determines whether parameter n is a perfect number.

    (b) Write a program that prints all the perfect numbers between 1 and 1000. Print the factors of each perfect number to confirm that the number is indeed perfect. You should use the function $Perfect(n)$ that you implemented in (a).

Perfect numbers between 1 and 1000: 6 = 1 + 2 + 3 28 = 1 + 2 + 4 + 7 + 14 ...

1부터 1000까지의 완전수를 출력하고 그 완전수를 약수들의 합으로 나타내는 프로그램이다.
int형 변수 num을 인자로 받아서 완전수인지 아닌지 판별하는 함수를 구현했다. 1000까지의 완전수 중에서 홀수는 없으므로 num을 2로 나눈 나머지가 1이라면 홀수이므로 완전수가 아니다. 크기가 1001인 int형 배열을 선언하여 약수들을 저장하고 약수들의 합을 저장하는 sum, num을 제외한 약수의 개수를 세어주는 cnt1, cnt2 변수를 선언했다.
for문을 통하여 1부터 num-1까지 반복하며 num%i==0이라면 i가 num의 약수이므로 arr[i]에 i값을 저장하고, cnt1을 하나 증가시키는 것을 반복하여 약수들을 뽑아낸다. 또한 이렇게 저장한 약수들을 sum에 더해주어서 약수들의 합을 구한다.
sum과 num이 같으면 완전수이므로 for문을 반복하여 arr[i]가 0이 아닐 때(num의 약수일 때) 약수들을 출력하고 cnt2를 하나 증가시킨다. cnt2가 cnt1보다 작을 때만 +기호를 출력하여 문제의 출력과 같이 완전수가 자신을 제외한 약수의 합으로 출력될 수 있도록 구현했다.

```
Perfect numbers between 1 and 1000:
6 = 1 + 2 + 3
28 = 1 + 2 + 4 + 7 + 14
496 = 1 + 2 + 4 + 8 + 16 + 31 + 62 + 124 + 248
```

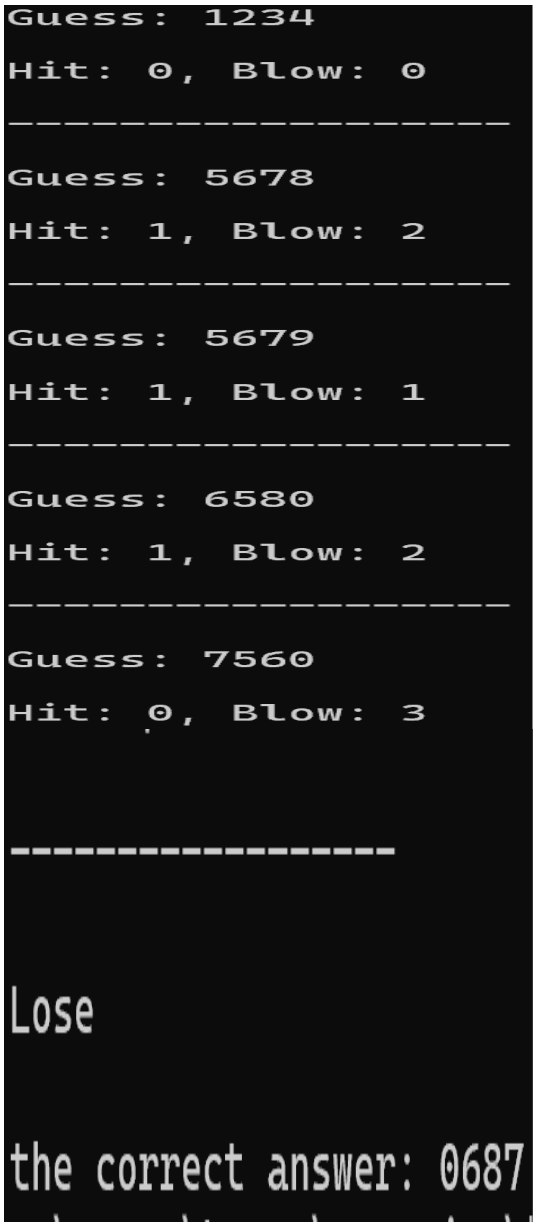
https://terms.naver.com/entry.naver?docId=3338423&cid=47324&categoryId=47324-네이버 지식백과 (완전수)
자기 자신을 제외한 약수들의 합이 자신과 같은 수를 완전수라고 부른다는 것을 몰랐는데 이번 과제를 통해서 알게 되었고, 아직 더 큰 완전수나 10^1500보다 작은 홀수인 완전수는 없다는 것을 완전수를 찾아보다가 알게 되었다.

15. Write the following program. 'Hit & Blow' game is similar to 'Number Baseball' game. When the game start, the program sets four random numbers among 0 to 9, not overlapping. These numbers are not displayed. Whenever you guess and input numbers, you get hints. If there are the same numbers in the same position between random numbers and yours, the program displays "Hit". And if in wrong position, the program displays "Blow". Within given 5 chances, if you obtain correct numbers, this program prints "Win". If you lose, this program prints "Lose" with the correct answer. (The correct answer should be different each time you run it.)

```
Guess: 1234

Hit: 1, Blow: 1

------------------

Guess: 3456
```

중복되지 않는 랜덤한 값 4개를 할당하여 그 숫자를 맞추는 게임을 구현하는 프로그램이다.
랜덤한 값과 입력한 값의 숫자가 맞으면 blow, 자리까지 맞다면 hit가 증가한다. 따라서 hit가 4개면 모두 맞춘 것이므로 게임에서 이기게 되고 5번의 기회가 주어진다. 게임에저 지게되면

맞추지 못한 랜덤한 값이 출력된다.
srand에 time(NULL)을 시드로 넣어서 매 실행마다 랜덤한 값을 출력하도록 했다. for 문을 통해서 i는 0부터 4까지 반복하며 값을 할당하고 0~9까지의 값을 할당해야 하므로 %10을 취해주었다. 또한 랜덤 값은 중복이 아니어야 하므로 for문을 하나 더 넣어서 j=0부터 i-1까지 반복하여 중복검사를 진행한다. arr[i]와 arr[j]가 같으면 i를 감소시키는데, 이렇게 되면 반복문이 다시 i를 할당하므로 중복이 나오지 않을 때 까지 반복하므로 중복된 랜덤 값이 나오지 않게 된다.
입력받는 숫자는 %연산자와 /연산자를 사용하여 1의 자리를 배열에 3, 2, 1, 0인덱스의 순서로 저장하도록 했고 이렇게 되면 랜덤한 입력과 같은 순서로 숫자가 저장된다. 이중반복문을 선언하여 랜덤한 값이 저장된 arr1[i]와 입력한 값이 저장된 arr2[j]가 같으면 blow가 1 증가하고, i와 j가 같으면 자리까지 같아지므로 hit가 증가된다. 이런 방식으로 구현하면 blow에는 hit까지 중복으로 센 값이 있으므로 blow에서 hit를 빼주어야한다.



원래 중복검사 같은 방법에서는 while문을 사용하여 조건이 맞을 때 까지 반복하여 중복검사를 진행하는 방식을 사용했는데, 이번 과제의 15번 문제에서는 랜덤으로 할당한 값이 중복되지 않기 위해서 이중반복문의 안쪽 반복문에서 i에 접근하여 하나씩 줄이는 방식을 처음 사용해보았다. 훨씬 코드가 간단해지고 다른 사람들뿐 아니라 내가 볼 때도 가독성이 좋은 코드인 것 같아서 나중에도 유용하게 사용할 수 있을 것 같다.