1. **7장 온라인 강의를 다 들은후 7장 연습문제중 (pp.350-351) 9, 10, 11, 13, 19, 20번 문제를 푸시오**

9.

a.

b.

c.

d.

e.

f.

10.

a.

b.

c.

d.

e.

f.

11.

* or, xor

<expr> -> <expr> or <or\_expr>

| <expr> xor <or\_expr>

| <or\_expr>

* and

<or\_expr> -> <or\_expr> and <and\_expr>

| <or\_expr>

* =, /=, <, <=, >=, >

<and\_expr> -> <and\_expr> = <expr>

| <and\_expr> /= <expr>

| <and\_expr> < <expr>

| <and\_expr> <= <expr>

| <and\_expr> >= <expr>

| <and\_expr> > <expr>

| <expr>

* -(unary)

<expr> -> -<unary\_expr>

| <unary\_expr>

* +, -, &, mod

<unary\_expr> -> <unary\_expr> + <term>

| <unary\_expr> - <term>

| <unary\_expr> & <term>

| <unary\_expr> mod <term>

| <term>

* \*, /, not

<term> -> <term> \* <factor>

| <term> / <factor>

| not <term>

| <factor>

<factor> -> <operand>

| ( <expr> )

<operand> -> a

| b

| c

| d

| e

13.

a.

b.

19.

a.

b.

20.

JAVA는 evaluation을 진행할 때 우선순위와 연관성에 어긋나지 않는다면 left to rigth으로 진행한다. Common language에서 associative는 left to right evaluation을 진행하므로 따라서 associative language이다. 그리고 이것을 JAVA expression evaluation이라 부르며, arithmetical evaluation과 동일한 의미를 가지고 있다.

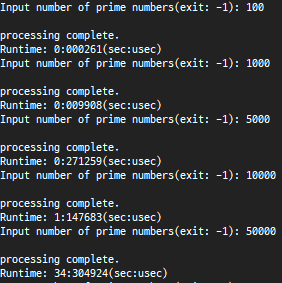
1. **N 값을 입력으로하여 N 개의 소수(prime number)를 출력하는 C언어 프로그램을 작성하고,  N이 각각 100, 1000, 5000, 10000 일때 프로그램의 실행시간을 측정하여 보시오**

* 소스코드

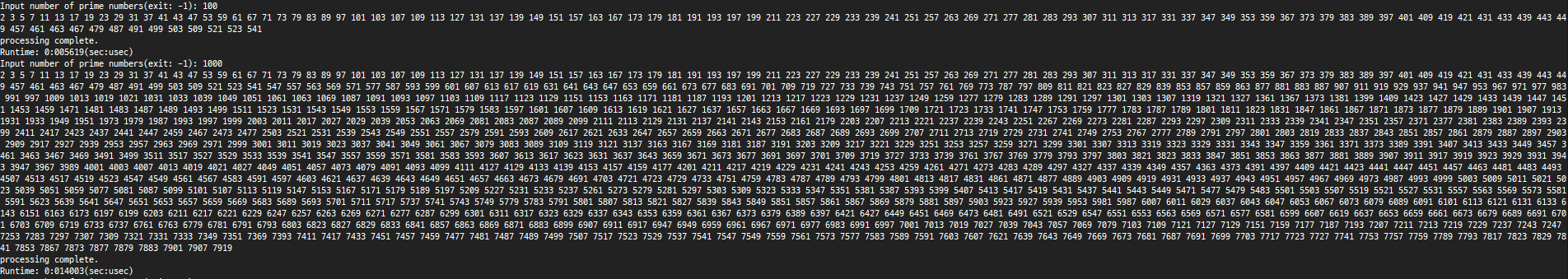
|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  #include <sys/time.h>  #define DEBUG  #define SECOND\_TO\_MICRO 1000000  void ssu\_runtime(struct timeval \*begin\_t, struct timeval \*end\_t);  int main(void)  {  int input;  int num; // 소수 판별 대상  int denom; // 나눌 수  struct timeval begin\_t;  struct timeval end\_t;  while (1) {  printf("Input number of prime numbers(exit: -1): ");  scanf("%d", &input);  if(input < 0)  break;  num = denom = 2;  gettimeofday(&begin\_t, NULL); // 측정 시작  for (int i = 0; i < input; i++) // 출력할 소수의 개수 만큼 반복  while (1) {  if (num % denom == 0) // 나누었을 때 나머지가 0일 경우  if (num == denom) { // 소수일 경우 (자신 이전에 나눌 수 있는 수가 없음)  #ifdef DEBUG  printf("%-d ", denom); // 소수 출력  #endif  num++; // 다음 수 탐색  denom = 2; // denom 초기화  break; // 다음 소수 탐색  } else { // 합성수일 경우 ( 자신 이전에 나눌 수 있는 수가 있음)  num++; // 다음 수 탐색  denom = 2; // denom 초기화  }  else denom++; // 모든 denom 탐색  }  printf("\nprocessing complete.\n");  gettimeofday(&end\_t, NULL);  ssu\_runtime(&begin\_t, &end\_t);  }  }  void ssu\_runtime(struct timeval \*begin\_t, struct timeval \*end\_t) // 실행시간 출력  {  end\_t->tv\_sec -= begin\_t->tv\_sec;  if(end\_t->tv\_usec < begin\_t->tv\_usec){  end\_t->tv\_sec--;  end\_t->tv\_usec += SECOND\_TO\_MICRO;  }  end\_t->tv\_usec -= begin\_t->tv\_usec;  printf("Runtime: %ld:%06ld(sec:usec)\n", end\_t->tv\_sec, end\_t->tv\_usec);  } |

[코드 1] main.c

* 실행 결과 – 출력 구문 제외



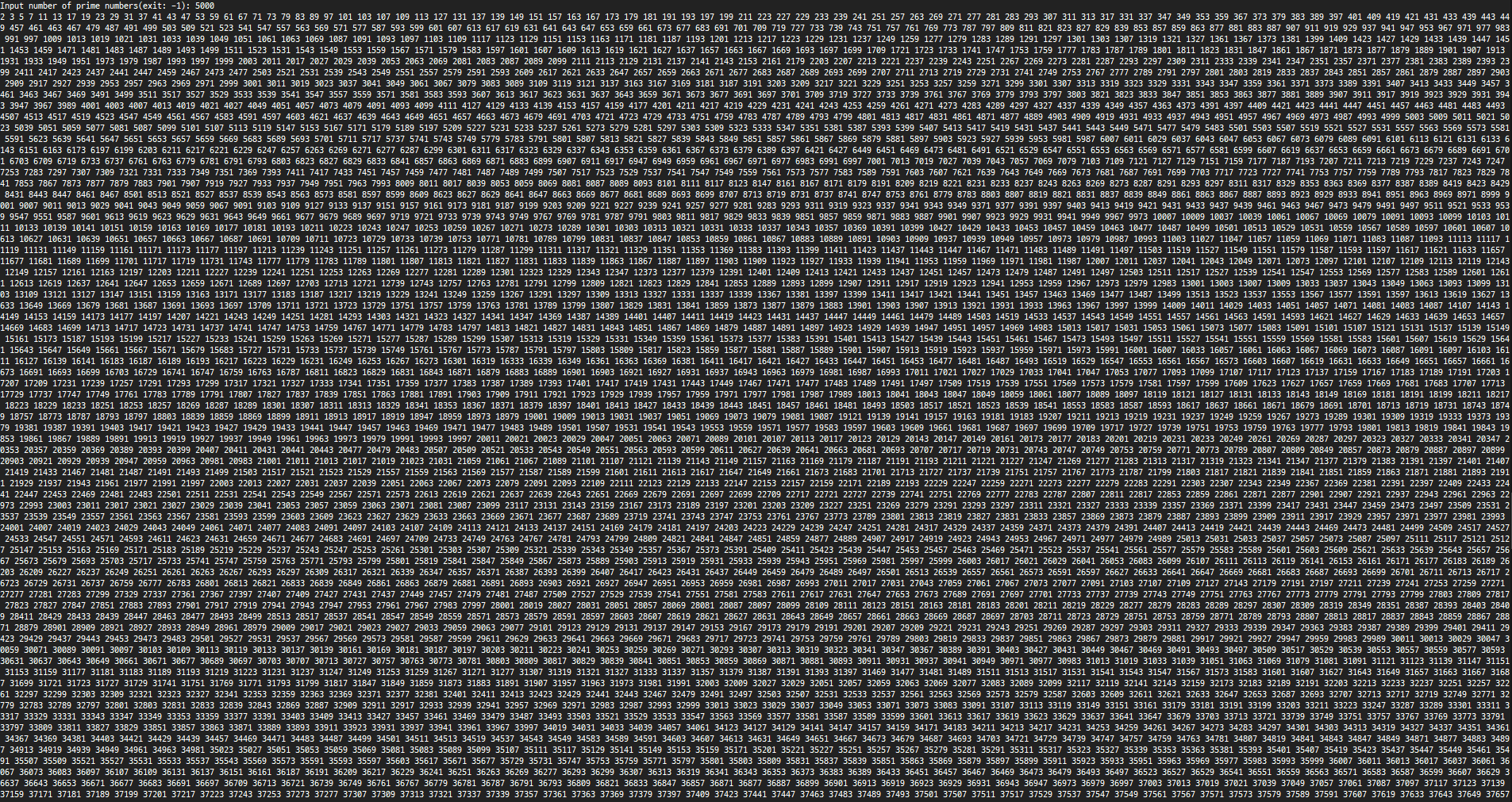
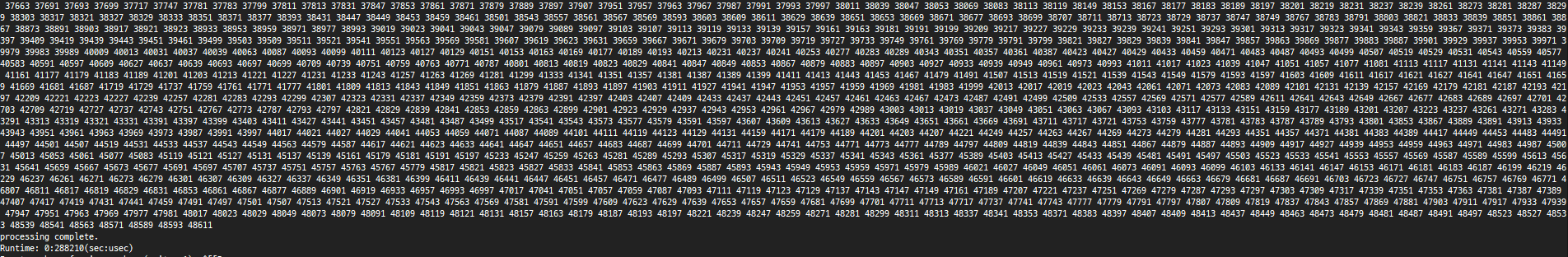
* 실행 결과 – 출력 구문 포함
  + N = 100, 1000



[그림 1] 좌:100, 우:1000

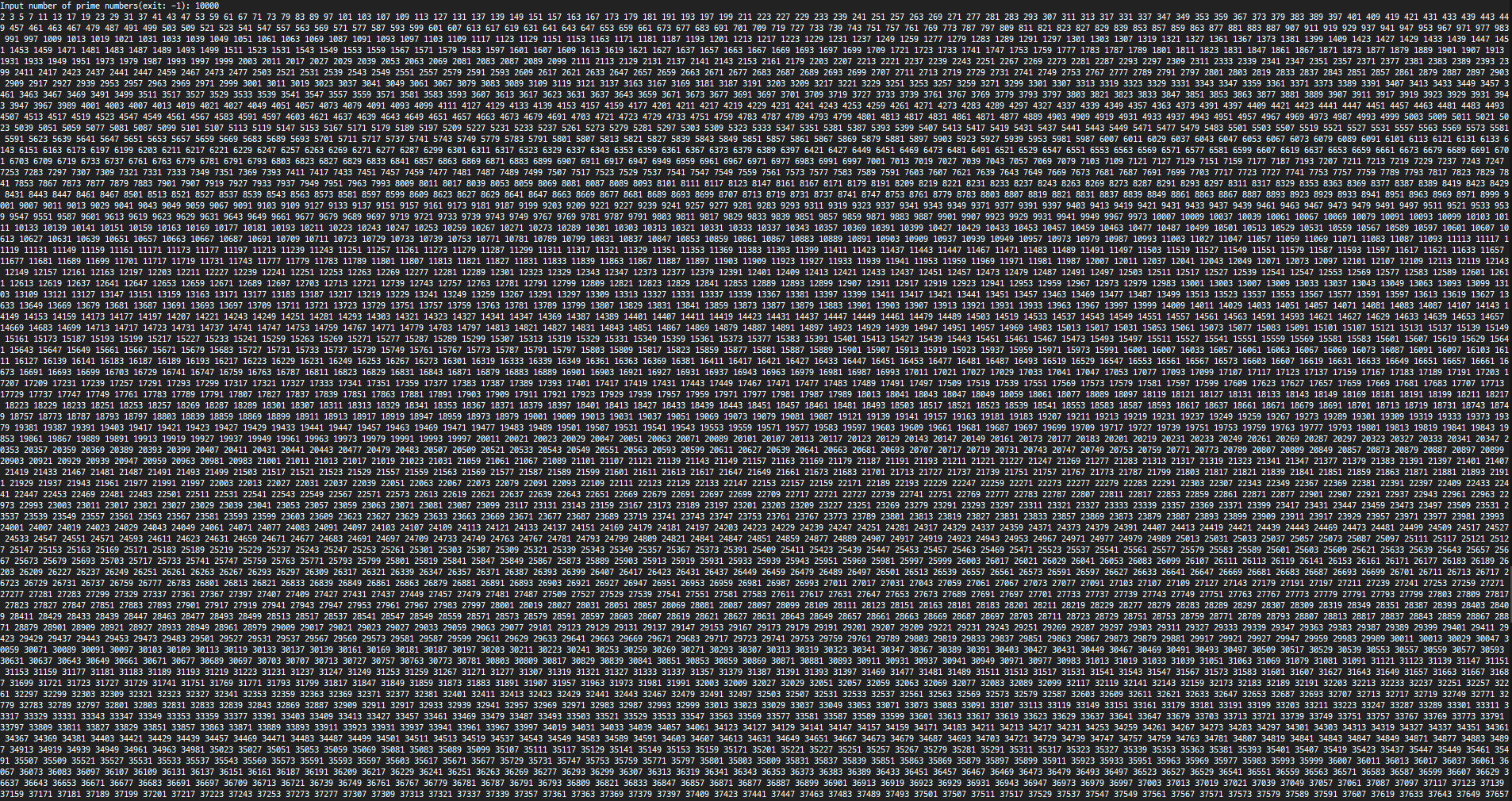
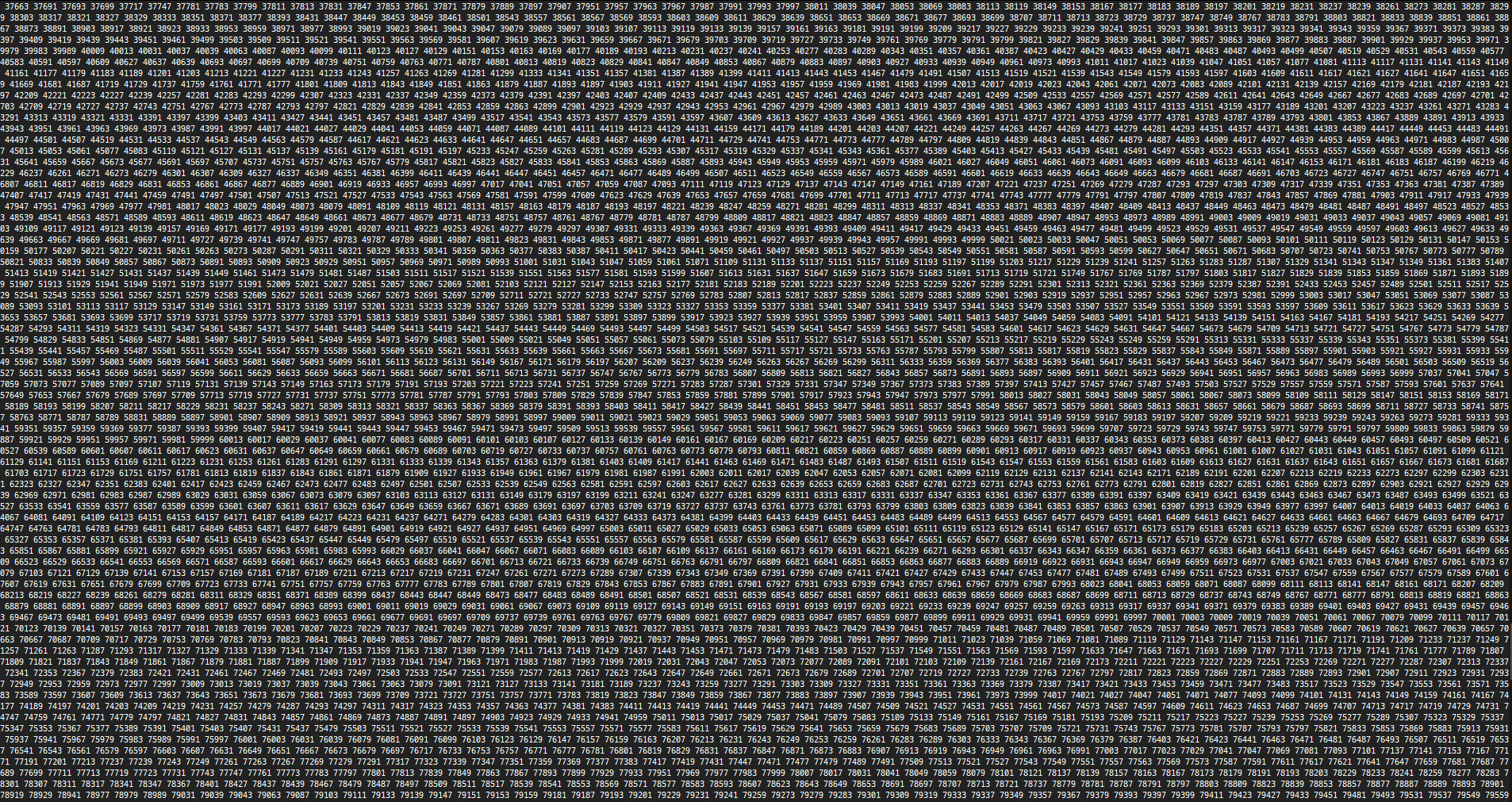
* + N = 5000



[그림 2] N = 5000

* + N = 10000



[그림 3] N = 10000