

## <1번 문제>

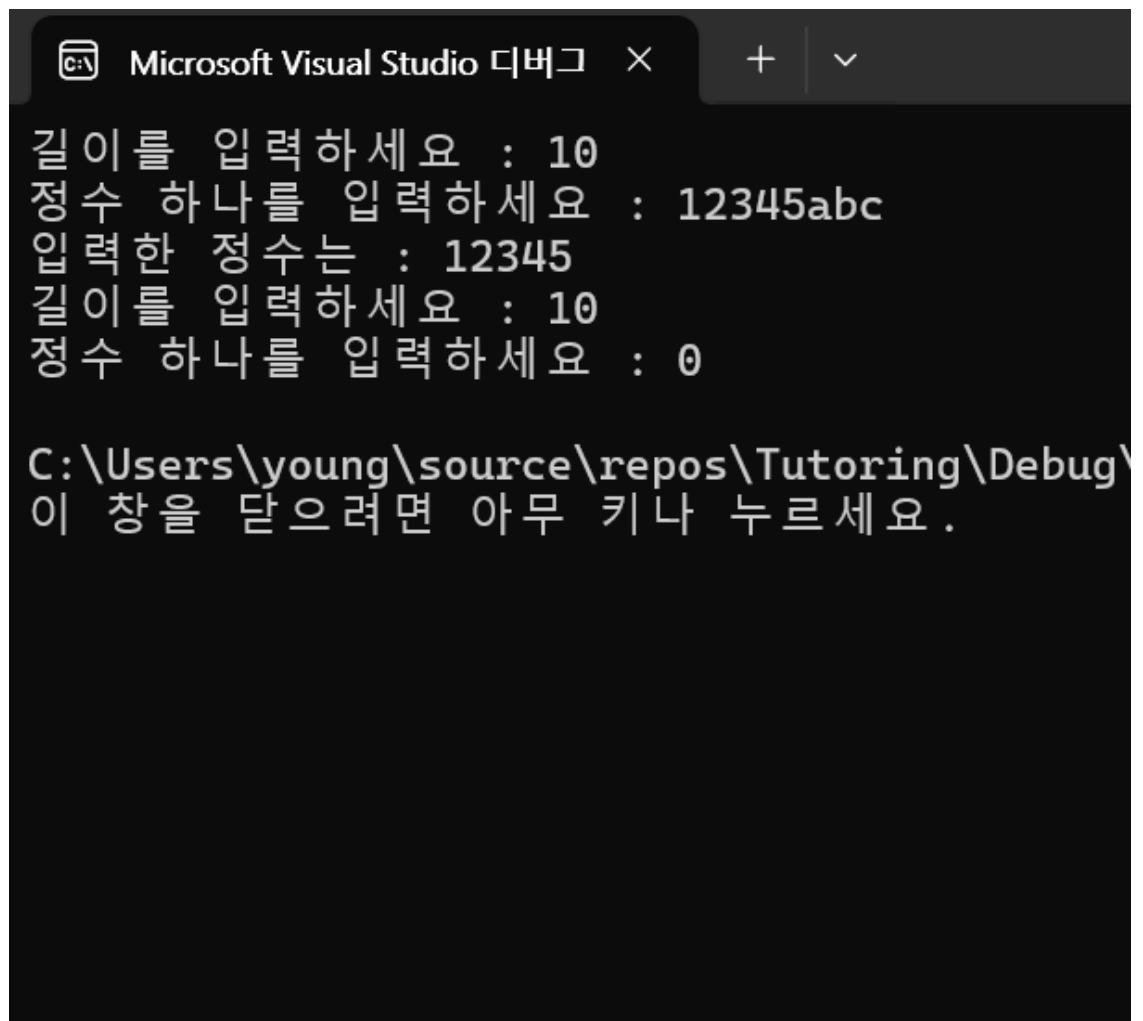
사용자가 "12345" 대신에 "12345abc"와 같이 문자를 추가로 입력하더라도, 숫자만이 정상적으로 출력되도록 프로그램을 작성하세요. (0만 입력 시 종료)

### <변수>

1. n : 문자열의 길이를 입력받는 변수
2. char\* input : 문자열을 입력받기 위해 동적할당한 변수

### <작동 과정>

1. 문자열의 길이만큼 동적할당 하기 위해 변수를 입력받음.
2. 변수 입력받은 후에 변수의 크기만큼 문자열 input 동적할당.
3. Input 값이 '0'이라면 while문 벗어나도록 break;
4. 그 외에는 아스키코드가 48 ~ 57 사이일 때만 값을 출력함.



```
Microsoft Visual Studio 디버그 × + v

길이를 입력하세요 : 10
정수 하나를 입력하세요 : 12345abc
입력한 정수는 : 12345
길이를 입력하세요 : 10
정수 하나를 입력하세요 : 0

C:\Users\young\source\repos\Tutoring\Debug\
이 창을 닫으려면 아무 키나 누르세요.
```

## <2번 문제>

20개의 난수를 생성하고 (-100~100), 병합 정렬 알고리즘을 사용하여 정렬

### <변수> - main 함수

1. int arr 배열 : 20개의 난수를 입력받기 위한 배열.

### <변수> - mergesort 함수

1. int \*arr : main 함수에 있는 배열을 가지고 오기 위한 포인터
2. int left : 배열의 왼쪽 끝을 저장하기 위한 변수
3. int right : 배열의 오른쪽 끝을 저장하기 위한 변수
4. int mid : left와 right의 중간 값을 저장하는 변수

### <변수> - merge 함수

1. int \*arr : main 함수에 있는 배열을 가지고 오기 위한 포인터
2. int left : mergesort의 함수의 left와 같은 역할
3. int right : mergesort의 함수의 right와 같은 역할
4. int mid : mergesort의 함수의 mid와 같은 역할

### <작동과정>

1. main 함수에서 20개의 난수를 생성하고 arr 배열에 저장함.
2. 배열을 1개의 크기로 만들기 위해서 mergesort 함수 사용함.
3. Mergesort 함수로 옮겨진 이후에 left와 right를 더한 값을 2로 나누어 mid 값을 구함.
4. Left < right일 때, 즉 배열의 크기가 1개 초과일 때에만 mergesort 재귀함수를 불러옴.
5. if문의 첫 번째 줄에서 배열의 왼쪽 10개를 나누고, 다시 재귀함수를 불러오고 왼쪽의 5개로 나누고, 다시 반복하여 3 -> 2 -> 1개로 배열을 나눔.
6. 이후 arr[0]과 arr[1]을 비교하기 위해 merge함수를 불러오고 temp 배열에 비교하고 난 값을 차례로 넣은 후 다시 arr 배열에 집어넣음.
7. 그 다음 arr[0], arr[1]과 arr[2]를 비교하기 위해 merge 함수를 불러오고 temp 배열에 비교한 값을 작은 순서대로 넣은 후에 다시 arr 배열에 집어넣음.
8. 이를 완전히 정렬될 때까지 반복함.
9. 완전히 정렬된 후 main 함수로 돌아와서 arr배열을 출력함.

```
Microsoft Visual Studio 디버그 × + v
-97 -93 -47 -46 -39 -36 -24 -14 -8 2 8 9 24 37 53 58 75 96 97 99
C:\Users\young\source\repos\Tutoring\Debug\Tutoring.exe(24768 프로세스)이(가) 0 코드로 인하
이 창을 닫으려면 아무 키나 누르세요.
|
```

## <3번 문제>

M by N 행렬 A와 A transpose 의 행렬곱을 계산하는 프로그램을 작성

### <변수>

1. M : 행을 입력받는 변수, N : 열을 입력받는 변수
2. Low : 행 또는 열 중에 작은 값을 저장, high : 행 또는 열 중에 큰 값을 저장
3. 2차원 배열 A, AT, Sum : 행렬을 저장하고, 행렬곱을 저장하기 위한 배열.

### <작동과정>

1. 행렬 A를 만들기 위해 M, N을 입력받고 동적할당을 통해  $M * N$ 의 2차원 배열 생성.
2. 행렬 AT를 만들기 위해 동적할당을 통해  $N * M$ 의 2차원 배열 생성.
3. 행렬 Sum을 만들기 위해 동적할당을 통해  $low * low$ 의 2차원 배열 생성.
4. 이후 행렬 A에 값을 집어넣고  $A[i][j]$  값을  $AT[j][i]$  값에 집어넣음.
5. 이후에 행렬 A, 행렬 AT 출력함.
6. 그리고 for문을 이용하여  $A*AT$ 를 하고  $Sum[i][j]$ 에 값을 집어넣음
7. Sum 행렬 출력

```
Microsoft Visual Studio 디버그 × + ▾

3 2

-10 3 3
5 1 4

-10 5
3 1
3 4

118 -35
-35 42

125 -25 -10
-25 10 13
-10 13 25

C:\Users\young\source\repos\Tutoring\Debug\Tutoring.exe
이 창을 닫으려면 아무 키나 누르세요.
|
```

## <4번 문제>

### <변수> - main

Int N : N \* N의 배열을 만들기 위한 변수

Double Sum : 최종 결과 값을 도출해내기 위한 변수

Arr: N \* N의 행렬을 만드는 배열

### <변수> - swap

Temp : 배열 값을 임시로 저장하기 위한 변수

### <작동 과정>

1. N값을 입력 받은 뒤에 N \* N의 배열을 만들기 위해 arr 배열을 N \* N으로 동적할당함.
2. (N, N) 번째의 항이 0이 되지 않도록 행을 바꿔줌

3. 이후 가우스 소거법을 사용하여 값을 계산함
4. 최종 결과 값을 도출해냄

```

Microsoft Visual Studio 디버그 × + ▾
13.00 14.00 15.00 16.00

1.00 2.00 3.00 4.00
0.00 -4.00 -8.00 -12.00
0.00 -8.00 -16.00 -24.00
13.00 14.00 15.00 16.00

1.00 2.00 3.00 4.00
0.00 -4.00 -8.00 -12.00
0.00 -8.00 -16.00 -24.00
0.00 -12.00 -24.00 -36.00

1.00 2.00 3.00 4.00
0.00 -4.00 -8.00 -12.00
0.00 0.00 0.00 0.00
0.00 -12.00 -24.00 -36.00

1.00 2.00 3.00 4.00
0.00 -4.00 -8.00 -12.00
0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00

1.00 2.00 3.00 4.00
0.00 -4.00 -8.00 -12.00
0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00

-0.000000
C:\Users\young\source\repos\Tutoring\Debug\Tutoring.exe(24676 프로세스)
이 창을 닫으려면 아무 키나 누르세요.

```

## <5번 문제>

숫자를 입력받으면 로마숫자로 변환하는 프로그램 작성

### <변수>

1. Num 배열 : 입력받은 숫자의 각 자리수를 입력받기 위한 변수
2. n : 로마숫자로 바꾸기 위한 숫자를 입력받는 변수
3. l : 로마숫자의 개수를 표현하고, l번째 arr 배열에 값을 집어넣기 위한 변수
4. cnt : arr 배열 동적할당을 위한 변수

5. arr : 로마숫자를 저장하기 위한 문자형 배열

### <작동 과정>

1. n을 입력받은 이후에 for문을 이용해 10으로 나눴을 때 나머지 값을 num 배열에 집어넣음.
2. Arr 배열을 선언하고 0으로 초기화함.
3. 각 자리수 별로 로마숫자를 구하기 위해  $i < 4$ 인 for문을 사용함
4. 천의 자리 수일 때에는 1 or 0일 때 밖에 없으므로 1일 때 로마숫자 M을 배열에 집어넣음.
5. 백의 자리 수 일 때에는 9 or 4 일 경우, 그 외의 경우로 나누고 5 이상일 경우, 그 외의 경우로 또 나눔.
6. 9 or 4일 경우에는 2칸 재할당을 하고 나서 로마숫자를 집어넣음.
7. 5 이상일 경우에는 1칸 재할당 후 D를 집어 넣고  $\text{num}[1] \% 5$ 의 값만큼 for문을 돌려 재할당 & C를 집어넣음
8. 5 이하일 경우에는  $\text{num}[1]$ 만큼 for문을 돌려 재할당 & C를 집어넣음.
9. 십의 자리수, 일의 자리수일 경우에도 5~8의 방식대로 진행함.
10. Arr 배열 출력함.

```
Microsoft Visual Studio 디버그 × + ▾
999
999 = 900 + 90 + 9 = CMXCIX, 6
C:\Users\young\source\repos\Tutoring\Debu
이 창을 닫으려면 아무 키나 누르세요.
|
```

## <6번 문제>

아다마르 행렬 구현 (재귀로 구현)

### <변수> - main 함수

1.  $n : 2^n * 2^n$  만큼의 행렬을 만들기 위한 변수
2.  $result : 2^n$  꼴의 형태를 저장하기 위한 변수
3.  $arr : \text{Hadamard}$  행렬을 저장하기 위한 2차원 배열

### <변수> - Hadamard 함수

1.  $x : arr$  배열을 좌표축으로 생각했을 때  $x$ 축을 구현하기 위한 변수
2.  $y : arr$  배열을 좌표축으로 생각했을 때  $y$ 축을 구현하기 위한 변수

### <작동 과정>

1.  $n$ 번째 Hadamard 행렬을 구현하기 위해  $n$  값을 입력받음.
2.  $2^n$  꼴을 구현하기 위해  $result$  변수 선언 후  $2^n$  값을 집어넣음.

3. arr 배열을 선언하고  $2^n * 2^n$ 만큼 동적할당함.
4. arr[0][0]은 무조건 1이므로 1을 넣어줌.
5.  $n = 0$ 일 때에는 그대로 출력하면 되므로  $n \neq 0$ 일 때에만 Hadamard 함수 실행.
6. Hadamard 함수에서는 result의 값을  $2^{(n-1)}$ 으로 저장함..
7.  $n == 1$ 일 때는  $(x + 1, y)$ ,  $(x, y + 1)$ 엔  $(x, y)$ 의 값을 그대로 넣고  $(x + 1, y + 1)$ 일 땐  $(x, y)$ 의 값의  $(-1)$ 을 곱한 값을 넣어줌.
8. 그 외의 경우엔  $(x, y)$ 의 값을  $(x + result, y)$   $(x, y + result)$ 에 집어 넣고  $(x, y)$ 의  $-1$ 을 곱한 값을  $(x + result, y + result)$ 에 집어 넣음
9. 그 후엔 재귀함수를 이용하여  $2^{(n-1)}$  크기의 행렬을  $(x, y)$ ,  $(x + result, y)$   $(x, y + result)$ ,  $(x + result, y + result)$ 에서 만듦.A.B.C.

```

Microsoft Visual Studio 디버그
4
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1
1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1
1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 1
1 1 1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1
1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1
1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 1
1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 -1
1 1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1
1 -1 1 -1 1 -1 1 -1 -1 1 -1 1 -1 1 -1
1 1 -1 -1 1 1 -1 -1 -1 -1 1 1 -1 -1 1
1 -1 -1 1 1 -1 -1 1 -1 1 1 -1 -1 1 -1
1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1
1 -1 1 -1 -1 1 -1 1 -1 1 -1 1 1 -1 -1
1 1 -1 -1 -1 -1 1 1 -1 -1 1 1 1 1 -1
1 -1 -1 1 -1 1 1 -1 -1 1 1 -1 1 -1 1

C:\Users\young\source\repos\Tutoring\Debug\Tutoring.
이 창을 닫으려면 아무 키나 누르세요.
|

```