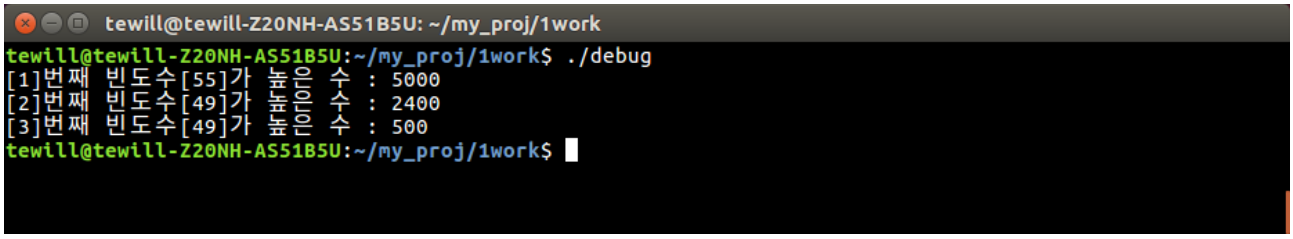
임베디드 애플리케이션 분석

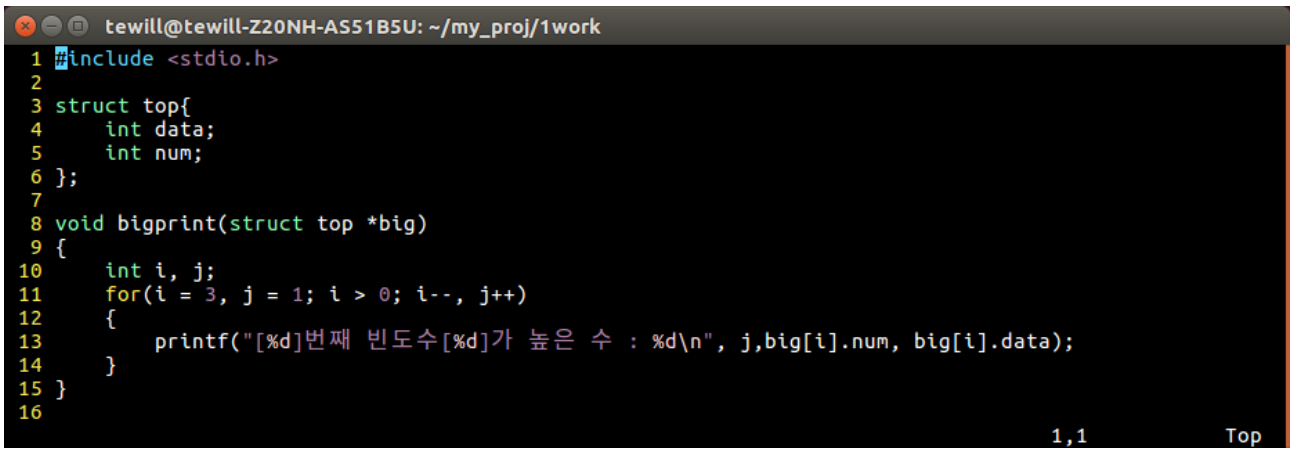
1. 이것이 없으면 사실상 C 언어를 사용할 수 없다.
C 언어에서 함수를 호출하기 위해서도 이것은 반드시 필요하다. 이와 같은 이유로 운영체제의 부팅 코드에서도이것을 설정하는 작업이 진행되는데 이것은 무엇일까 ?

- 메모리[스택 영역 / 힙 영역 / 데이터 영역 / 코드 영역]

2. 배열에 아래와 같은 정보들이 들어있다.
2400, 2400, 2400, 2400, 2400, 2400, 2400, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 2400, 2400, 2400, 2400, 2400, 2400, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 2400, 2400, 2400, 2400, 2400, 5000, 1, 2, 3, 4, 5, 5000, 5000, 500, 500, 500, 500, 500, 1, 2, 3, 4, 5, 500, 500, 500, 500, 500, 500, 500, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5,1,2,3, 4, 5,6, 7, 8, 9, 10,11, 12,13,14,15,16,17,18,234, 345, 26023, 346,345,234, 457, 3, 1224, 34,646, 732, 5,4467,45,623, 4, 356, 45,6, 123, 3245, 6567, 234,567, 6789,123, 2334, 345, 4576, 678, 789, 1000, 2400, 2400,2400, 2400, 2400, 2400, 1,2,3, 4,5,1,2,3, 4, 5,2400, 2400, 2400, 2400,978,456,234756, 5000,5000, 5000,2400, 500, 5000, 2400, 500, 500, 500,500,500, 500, 1,2,3,4, 5,500, 500,500, 500, 500, 500,500, 500, 500, 500, 1,2,3,4,5,500, 500, 500, 5000, 2400, 5000, 5000, 5000, 5000, 5000, 5000, 5000, 5000,1,2,3,4,5,5000, 5000,5000, 5000, 2400 5000, 500, 2400, 5000, 5000, 5000, 5000, 5000,5000,5000, 5000,1,2,3, 4,5,1,2,3,4,5,5000, 5000, 5000, 5000, 5000, 1,2,3,4,5,5000, 5000, 5000,5000, 5000,234, 4564, 3243, 876,645,534,423,312,756,235, 75678, 2400, 5000, 500, 2400, 5000, 500,2400, 5000, 500, 2400, 5000, 500,2400, 5000, 500,2400, 5000, 500,2400, 5000, 500,2400, 5000, 500,2400, 5000, 500, 7, 8, 9, 6,7, 8, 9, 6,7, 8, 9, 6,7, 8, 9, 6,7, 8, 9, 6,7, 8, 9, 6,500,2400, 5000,500,2400, 5000,500,2400, 5000,500,2400, 5000,500,2400, 5000,500,2400, 5000
여기서 가장 빈도수가 높은 3 개의 숫자를 찾아 출력하시오! 함수에서 이 작업을 한 번에 찾을 수 있도록 하시오.
(찾는 작업을 여러번 분할하지 말란 뜻임)

-





tewill@tewill-Z20NH-AS51B5U: ~/my_proj/1work

```
16
17 void explor(int *num, int size)
18 {
19     struct top big[4] = {0};
20     int i, j;
21     struct top tmp;
22
23     big[0].num = 1;
24
25     for(i = 1; i < size; i++)
26     {
27         if(num[i-1] == num[i])
28         {
29             big[0].num++;
30         }else {
31             for(j = 0; j < 3; j++)
32             {
33                 if(big[j].num > big[j+1].num){
34
35                     tmp = big[j+1];
36
37                     big[j+1] = big[j];
38                     big[j] = tmp;
39                 }
40             }
41
42             big[0].data = num[i];
43             big[0].num = 1;
44         }
45     }
46
47     for(j = 0; j < 3; j++)
48     {
49         if(big[j].num > big[j+1].num){
50
51             tmp = big[j+1];
52
53             big[j+1] = big[j];
54             big[j] = tmp;
55         }
56     }
57
58     bigprint(big);
59 }
60
                                                    16,0-1        33%
```

tewill@tewill-Z20NH-AS51B5U: ~/my_proj/1work

```
80
81 int main(void)
82 {
83     int num[] = {2400, 2400,   2400, 2400, 2400, 2400, 2400, 1,     2,   3,    4,      5,    1,
     2,   3,    4,      5,    2400, 2400,  2400,   2400, 2400,   2400, 1,     2,   3,    4,      5,
     1,     2,   3,    4,      5,    2400, 2400, 2400, 2400, 2400, 5000,  1,     2,   3,    4,      5
   ,    5000, 5000, 500,  500,  500,  500,  500,   1,     2,   3,    4,      5,    500,    500,  500,
     500,  500,  500,  500,  1,     2,   3,    4,      5,    1,     2,   3,    4,      5,    1,
    2,   3,    4,      5,    1,     2,   3,    4,      5,    1,     2,   3,    4,      5,    1,     2
   ,   3,    4,      5,    1,     2,   3,    4,      5,    1,     2,   3,    4,      5,    1,     2,
     3,    4,      5,    1,     2,   3,    4,      5,    1,     2,   3,    4,      5,    1,     2,
   3,    4,      5,    1,     2,   3,    4,      5,    1,     2,   3,    4,      5,    1,     2,   3,
     4,      5,    1,     2,   3,    4,      5,    1,     2,   3,    4,      5,    1,     2,   3,
     4,      5,    1,     2,   3,    4,      5,    1,     2,   3,    4,      5,    6,    7,    8,
   9,    10,   11,    12,    13,   14,    15,   16,   17,   18,   234,  345,  26023, 346,    345,  2
   34,    457,  3,    1224, 34,   646,  732,  5,     4467,    45,   623,    4,    356,  45,   6,    1
   23,  3245, 6567,  234,   567,  6789,   123,  2334, 345,  4576, 678,  789,  1000,  2400, 2400,   24
   00, 2400, 2400, 2400, 1,     2,   3,    4,      5,    1,     2,   3,    4,      5,    2400, 2400,
   2400,  2400,   978,   456,  234756, 5000,   5000, 5000,   2400, 500, 5000, 2400, 500,  500,  500,
     500,   500,  500,    1,     2,   3,    4,      5,    500,  500,  500,  500,  500,  500,  500, 50
   0, 500, 500, 1,     2,   3,    4,      5,    500, 500, 500, 5000, 2400, 5000, 5000, 5000, 5000, 50
   00, 5000, 5000,  5000,    1,     2,   3,    4,      5,    5000, 5000,   5000, 5000, 2400, 5000, 50
   0, 2400, 5000, 5000, 5000, 5000, 5000,   5000, 5000,   1,     2,   3,    4,      5,    1,
     2,   3,    4,      5,    5000, 5000, 5000, 5000, 5000, 1,     2,   3,    4,      5,    5000, 50
   00,  5000,   5000, 5000,   234,  4564, 3243, 876,  645,  534,  423,   312,   756,  235,    75678,
   2400, 5000, 500, 2400, 5000, 500, 2400, 5000, 500, 2400, 5000, 500, 2400, 5000, 500, 2400, 5000, 5
   00, 2400, 5000, 500, 2400, 5000, 500, 2400, 5000, 500, 2400, 5000, 500, 7, 8, 9, 6, 7, 8, 9, 6, 7,
    8, 9, 6, 7, 8, 9, 6, 7, 8, 9, 6, 7, 8, 9, 6,  500, 2400, 5000,  500, 2400, 5000,  500, 2400, 5000
   ,  500, 2400, 5000,  500, 2400, 5000,  500, 2400, 5000};
84     int size;
85     size = sizeof(num)/sizeof(int);
86
87     line_up(num, size);
88     explor(num, size);
89     return 0;
90 }
                                                    90,1          Bot
```

3. 12 비트 ADC 를 가지고 있는 장치가 있다.
보드는 12 V 로 동작하고 있고 ADC 는 -5 ~ 5 V 로 동작한다.
ADC 에서 읽은 값이 2077 일 때 이 신호를 디지털 관점에서 재해석하도록 프로그래밍 한다.

-

4. goto 의 정체성은 ?
전세계 각지의 천재들이 모여서 개발하는 리눅스 운영체제 코드에는 엄청 많은 양의 goto 가 사용되고 있다.
goto 를 도대체 왜 사용해야만 할까 ?

- goto 를 사용하지 않을 경우, 파이프라인 손실이 많아질 수 있기 때문이다. 파이프 라인 손실이란 call 등 jmp 가 사용될 때 발생하는데, jmp 기능이 실행 될 때 미리 진행되고 있던 것들을 버리고 다시 push 에서 시작되는 것을 이야기 한다. goto 에 대한 예를 보면, 여러 반복문 속에서 밖으로 나가기 위해서는 반복문 숫자만큼 break 를 해야하는데, goto 를 이용하면 한번에 원하는 곳으로 이동이 가능하다.

5. 포인터 크기에 대해 알고 있는대로 기술하시오.

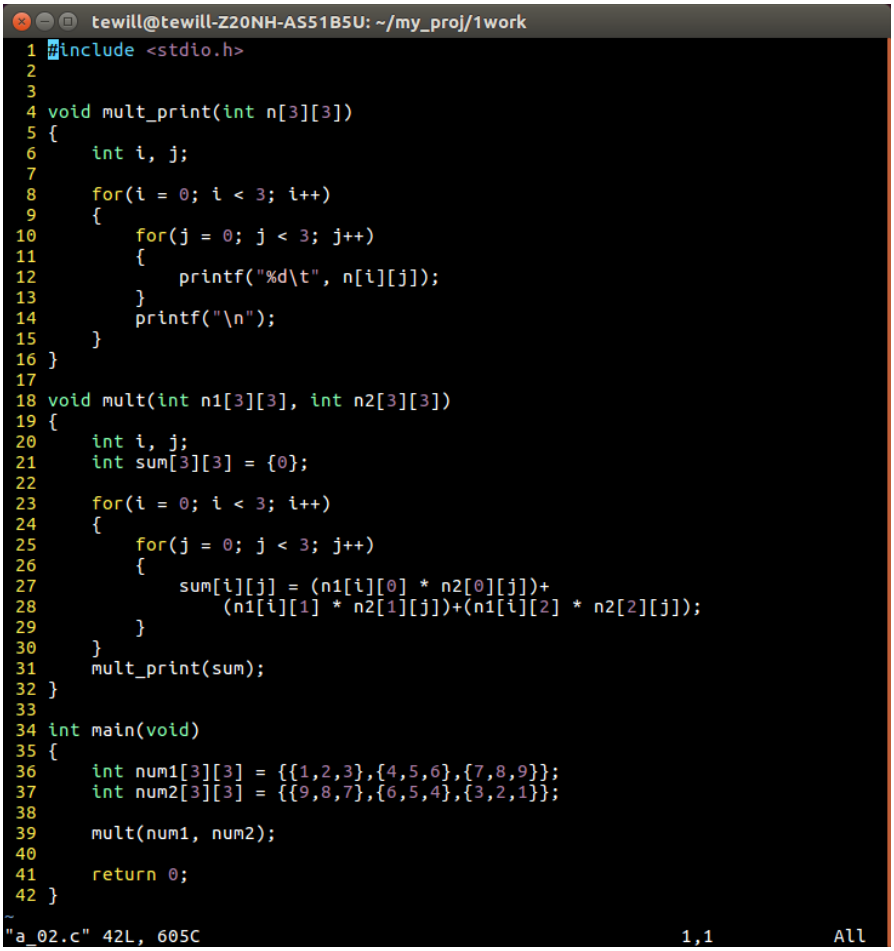
- 포인터는 모든 주소를 담을 수 있어야 하기 때문에 cpu 의 범용 레지스터의 크기로 포인터의 크기가 정해진다. 즉, 32 비트 시스템인지 64 비트 시스템인지에 따라 포인터의 크기가 4 바이트 혹은 8 바이트로 정해진다.

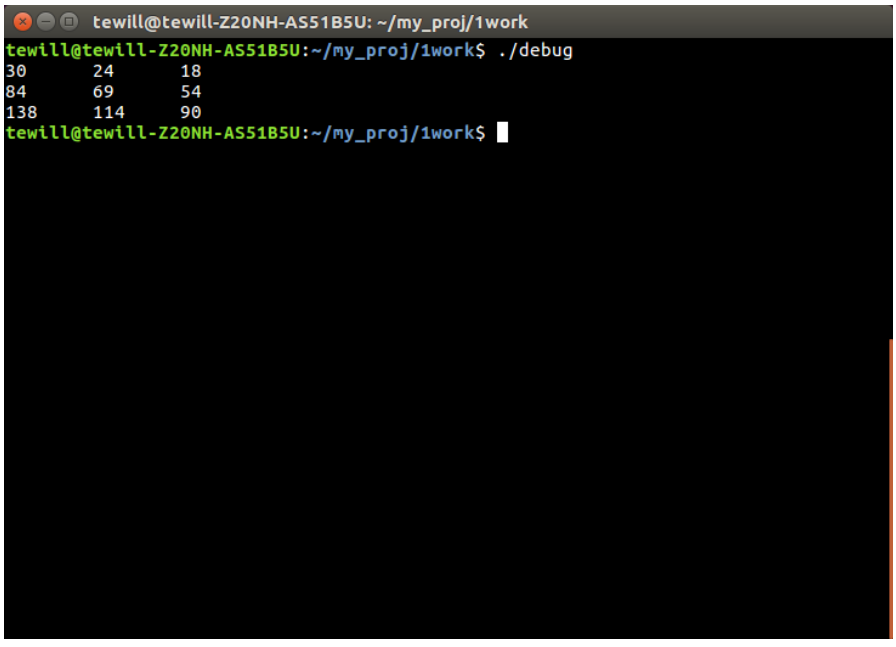
6. 위의 문장에서 Safety MCU 가 몇 번째 부터 시작하는지 찾아내도록 프로그래밍 해보자.
TI Cortex-R5F Safety MCU is very good to Real-Time System.
(이정도는 가볍게 해야 파싱 같은것도 쉽게 할 수 있다)
무능력자가 되지 않기 위해서라도 이정도는 가볍게 할 수 있어야 한다.

-

7. 이중 배열을 함수의 인자로 입력 받아 3 by 3 행렬의 곱셈을 구현하시오.

-





8. 다음 프로토타입을 기술하시오.
void (* signal(int signum, void (* handler)(int)))(int)

-
리턴 값 :void (*)(int)
이름 : signal
인자 : int signum, void (* handler)(int)

9. 다음 코드를 작성하라.
함수 포인터를 반환하고 함수 포인터를 인자로 취하는 함수의 주소를 반환하고

인자로 int 2 개를 취하는 함수를 작성하도록 한다.

(프로토타입이 각개 다를 수 있으므로 프로토타입을 주석으로 기술하도록 한다)

-

10. 파이프라인이 깨지는 경우 어떠한 이점이 있는지 기술하시오.

-

11. 아래 프로그램을 작성한다.
4x^2 + 5x + 1 을 1 ~ 3 까지 정적분하는 프로그램을 구현하도록 한다.

-

12. 아래 프로그램을 작성하시오.

함수 포인터를 활용하여 float 형 3 by 3 행렬의 덧셈과 int 형 3 by 3 행렬의 덧셈을 구현하도록 하시오.

-

tewill@tewill-Z20NH-AS51B5U: ~/my_proj/1work

```
 1 #include <stdio.h>
 2
 3
 4 void sumint(int (*n1)[3], int (*n2)[3])
 5 {
 6     int i, j;
 7
 8     for(i = 0; i < 3; i++)
 9     {
10         for(j = 0; j < 3; j++)
11             printf("%d\t",n1[i][j] + n2[i][j]);
12         printf("\n");
13     }
14
15 }
16
17 void sumfloat(float (*n1)[3], float (*n2)[3])
18 {
19     int i, j;
20
21     for(i = 0; i < 3; i++)
22     {
23         for(j = 0; j < 3; j++)
24             printf("%f\t",n1[i][j] + n2[i][j]);
25         printf("\n");
26     }
27
28 }
29
-- INSERT --                                                    1,19          Top
```

tewill@tewill-Z20NH-AS51B5U: ~/my_proj/1work

```
29
30 int main(void)
31 {
32     int n1[3][3] = {{1, 2, 3},{1, 2, 3},{1, 2, 3}};
33     int n2[3][3] = {{4, 5, 6},{4, 5, 6},{4, 5, 6}};
34
35     float f1[3][3] = {{1.1, 1.2, 1.3},{2.1 ,2.2, 2.3},{3.1, 3.2, 3.3}};
36     float f2[3][3] = {{1.1, 1.2, 1.3},{2.1, 2.2, 2.3},{3.1, 3.2, 3.3}};
37
38     void (*pint)(int *, int *);
39     pint = sumint;
40     pint(n1, n2);
41
42     void (*pfloat)(float *, float *);
43     pfloat = sumfloat;
44     pfloat(f1, f2);
45
46     return 0;
47 }
-- INSERT --                                                    47,2          Bot
```

tewill@tewill-Z20NH-AS51B5U: ~/my_proj/1work

```
tewill@tewill-Z20NH-AS51B5U:~/my_proj/1work$ ./debug
5       7       9
5       7       9
5       7       9
2.200000        2.400000        2.600000
4.200000        4.400000        4.600000
6.200000        6.400000        6.600000
tewill@tewill-Z20NH-AS51B5U:~/my_proj/1work$
```

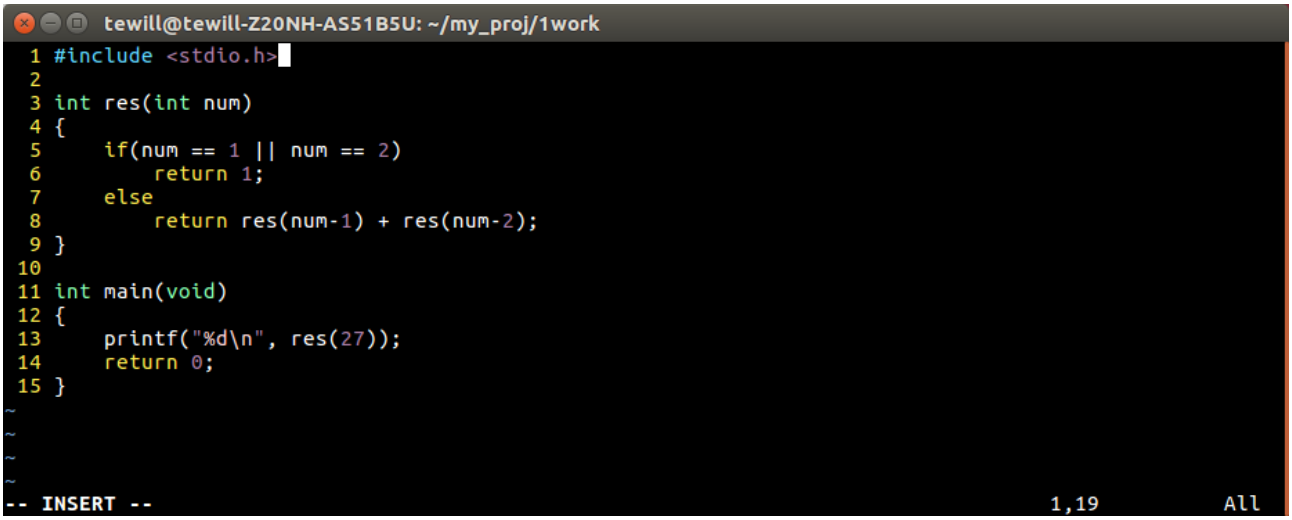
13. 아래 설명을 보고 프로그래밍 하시오.

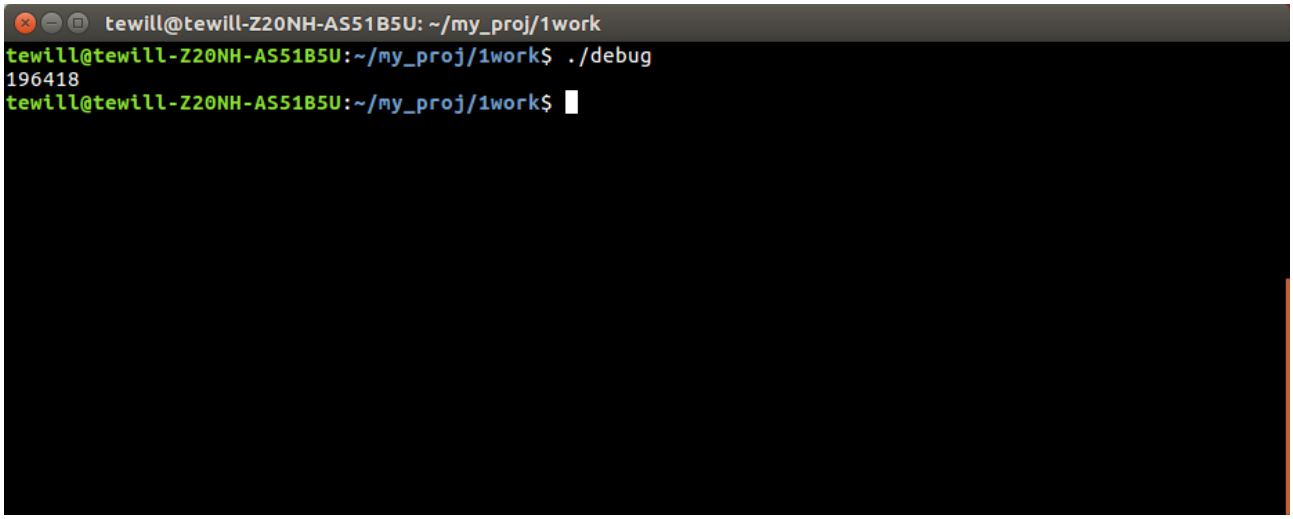
1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ... 형태로 숫자가 진행된다.

1 ~ 27 번째까지의 수들로 홀수들의 합을 하고 짝수들의 합을 구한다.

홀수들의 합 - 짝수들의 합의 결과를 출력하시오.

-



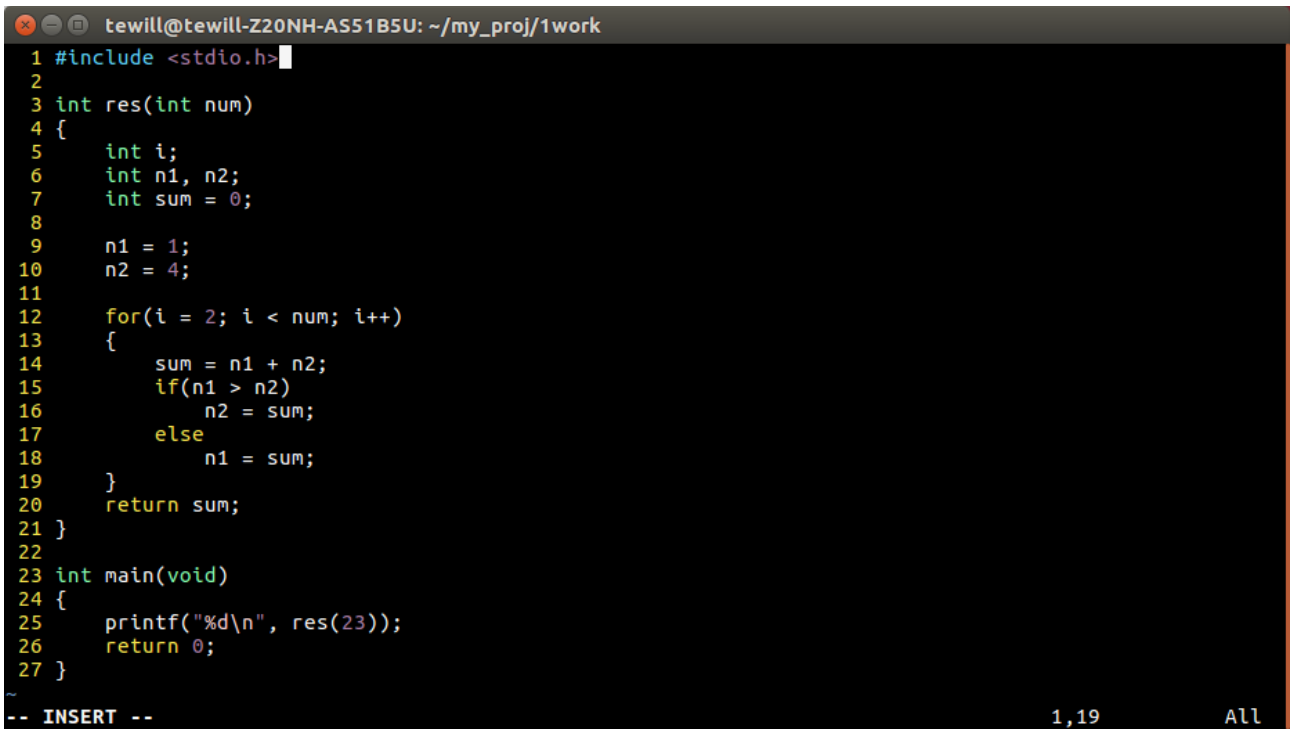


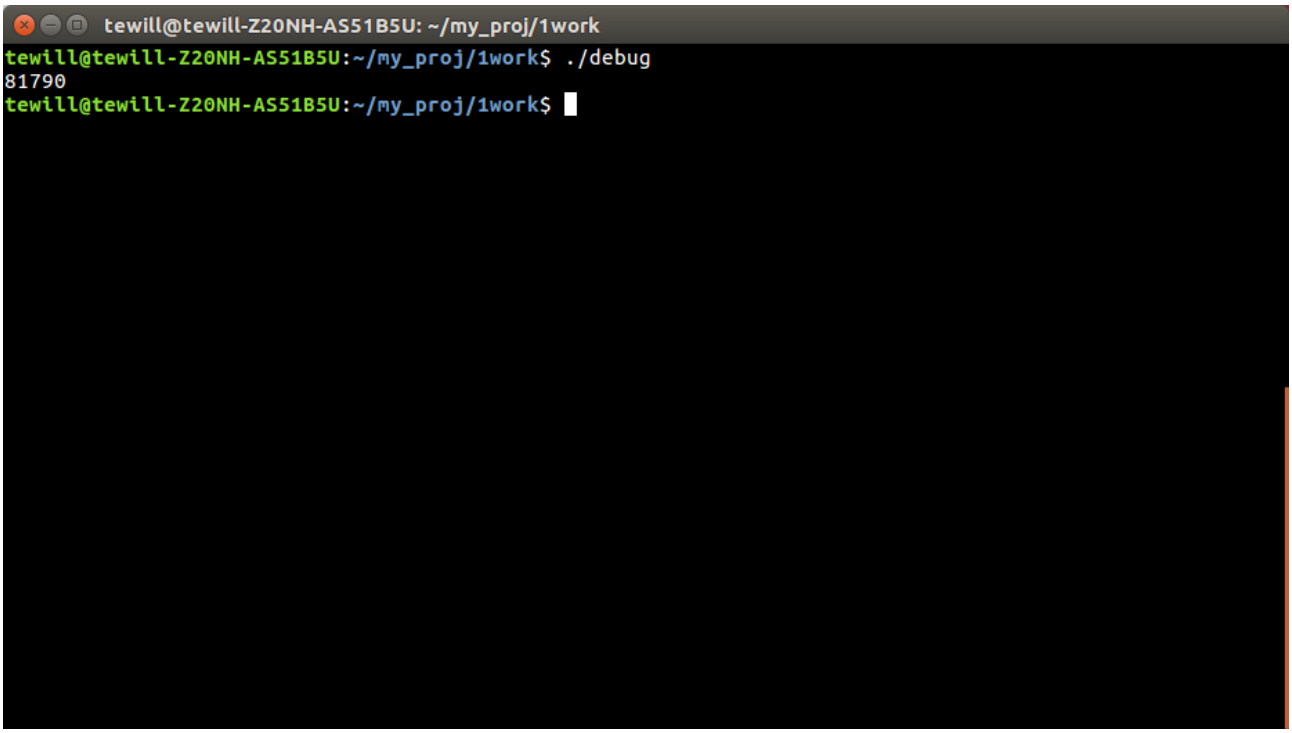
14. 아래 설명을 보고 프로그래밍 하시오.
1, 4, 5, 9, 14, 23, 37, 60, 97, ... 형태로 숫자가 진행된다.

23 번째 숫자는 무엇일까 ?

(프로그래밍 하시오)

-





15. Intel Architecture 와 ARM Architecture 의 차이점은 ?

- cisc 와 risc 의 차이가 있다. intel 의 경우 risc 를 사용하며, arm 의 경우 cisc 을 사용한다.
cisc 는 구조가 복잡하고 전력소모가 크지만, 적은 수의 명령어로 복잡한 프로그램을 만들 수 있다.

16. 네이버의 쓰레기같은 사다리 게임을 우리끼리 즐길 수 있는 것으로 대체하는 프로그램을 만들어보자.
우리반 학생들은 모두 25 명이다. 반 학생들과 함께 진행할 수 있는 사다리 게임을 만들도록 한다. 참여 인원수를 지정할 수 있어야하며 사다리 게임에서 걸리는 사람의 숫자도 조정할 수 있도록 만든다.

-

17. 아래 설명을 보고 프로그래밍 해보자.
아래와 같은 행렬을 생각해보자!

2 4 6

2 4 6

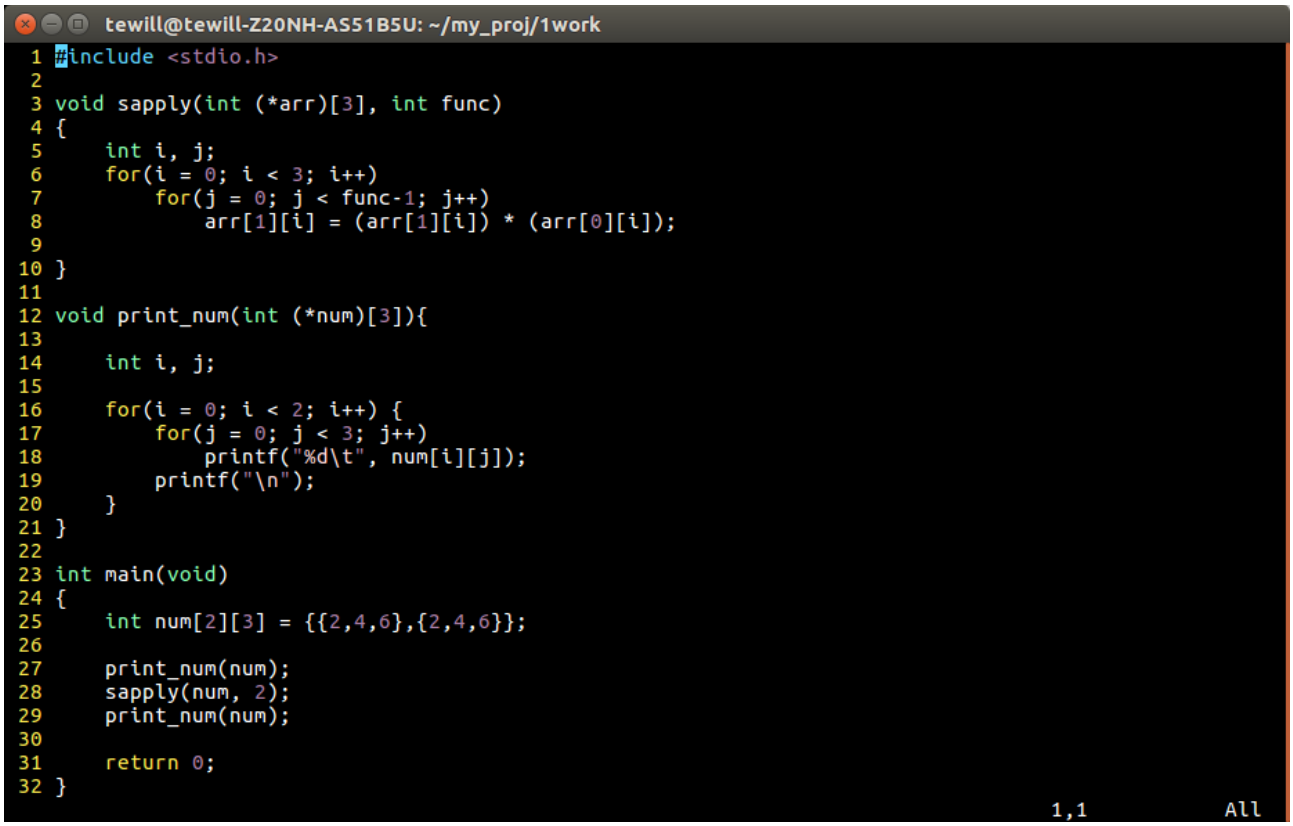
sapply(arr, func) 으로 위의 행렬을 아래와 같이 바꿔보자!

2 4 6

4 16 36

sapply 함수를 위와 같이 구현하라는 의미다.

(R 이나 python 같은 언어에서 지원되는 기법중 하나에 해당한다)




```c
#include <stdio.h>

void sapply(int (*arr)[3], int func)
{
    int i, j;
    for(i = 0; i < 3; i++)
        for(j = 0; j < func-1; j++)
            arr[1][i] = (arr[1][i]) * (arr[0][i]);

}

void print_num(int (*num)[3]){

    int i, j;

    for(i = 0; i < 2; i++) {
        for(j = 0; j < 3; j++)
            printf("%d\t", num[i][j]);
        printf("\n");
    }
}

int main(void)
{
    int num[2][3] = {{2,4,6},{2,4,6}};

    print_num(num);
    sapply(num, 2);
    print_num(num);

    return 0;
}
```

```
tewill@tewill-Z20NH-AS51B5U: ~/my_proj/1work
tewill@tewill-Z20NH-AS51B5U:~/my_proj/1work$ ./debug
2       4       6
2       4       6
2       4       6
4       16      36
tewill@tewill-Z20NH-AS51B5U:~/my_proj/1work$
```

18. 아래 설명을 보고 프로그램을 만들어보자.
char *str = "WTF, Where is my Pointer ? Where is it ?" 라는 문자열이 있다.

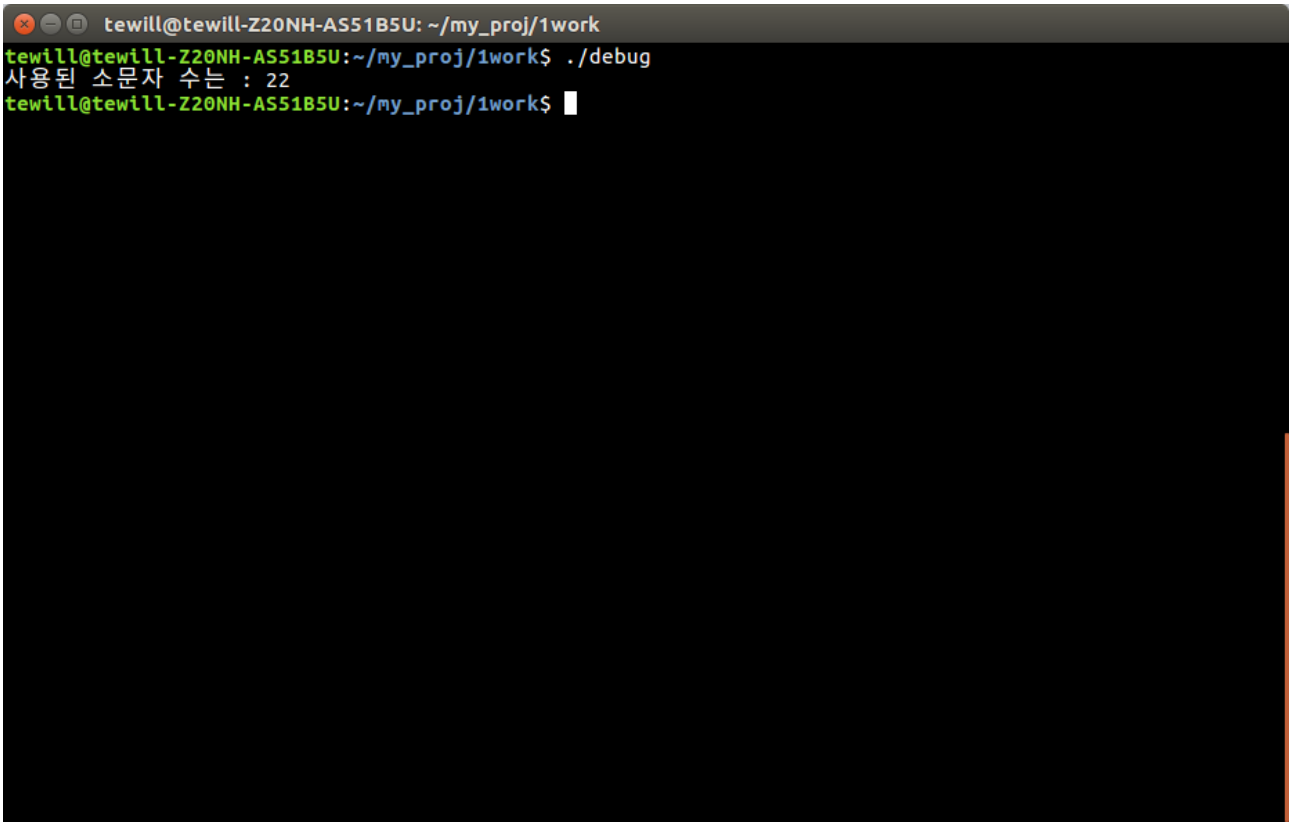
여기에 소문자가 총 몇 개 사용되었는지 세는 프로그램을 만들어보자!

-

tewill@tewill-Z20NH-AS51B5U: ~/my_proj/1work

```c
#include <stdio.h>

int res(char *str, int size)
{
    int i, num = 0;

    for(i = 0; i < size; i++)
    {
        if(str[i] >= 'a' && str[i] <= 'z')
            num++;
    }
    return num;
}

int main(void)
{
    char *str = "WTF,Where is my Pointer?Where is it?";
    int size = 0;

    for(size = 0; str[size] != NULL; size++);

    printf("사용된 소문자 수는 : %d\n", res(str, size));
        return 0;
}
```

```
~
~
~
~
~
~
~
~
"a_09.c" 24L, 413C                                            19,1          All
```



19. int *p[3] 와 int (*p)[3] 는 같은 것일까 ? 다른 것일까 ? 이유를 함께 기술하도록 한다.

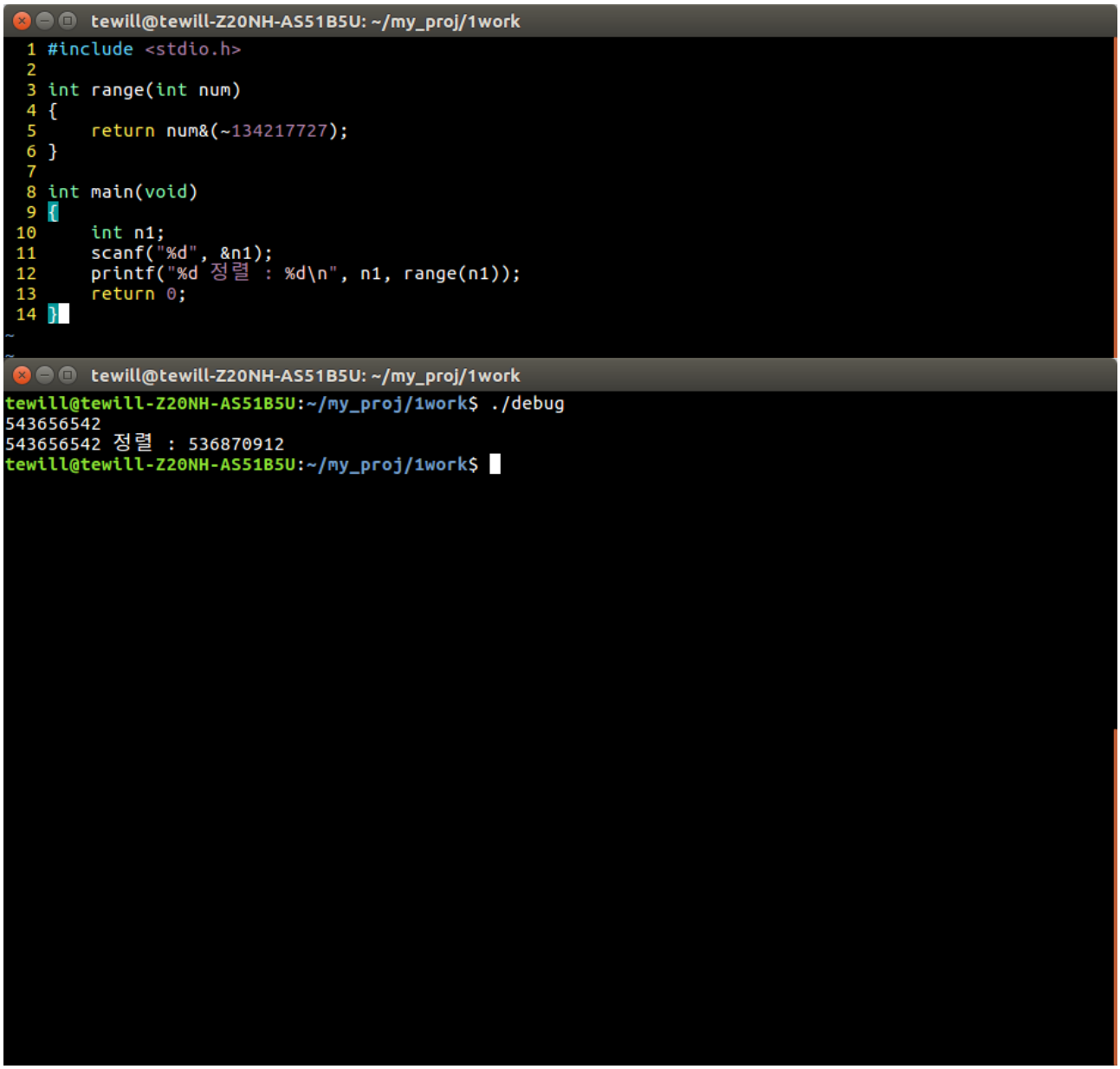
- int *p[3] 는 포인터로 된 배열로서, 포인터가 3 개라는 뜻으로 p[0], p[1], p[2] 로 구성되어 있다. 반면 int (*p)[3] 은 int 3 개만큼의 크기인 포인터를 나타낸다. 즉 12 바이트를 나타내는 포인터이다. 배열의 포인터라고 불린다. 풀어서 보면 ‘int *[3] p’ 형태이다.

20. 아래 설명을 보고 프로그램을 작성한다.
임의의 값 x 가 있는데, 이를 134217728 단위로 정렬하고 싶다면 어떻게 해야할까 ?

어떤 숫자를 넣던 134217728 의 배수로 정렬이 된다는 뜻임

(힌트 : 134217728 = 2^27)

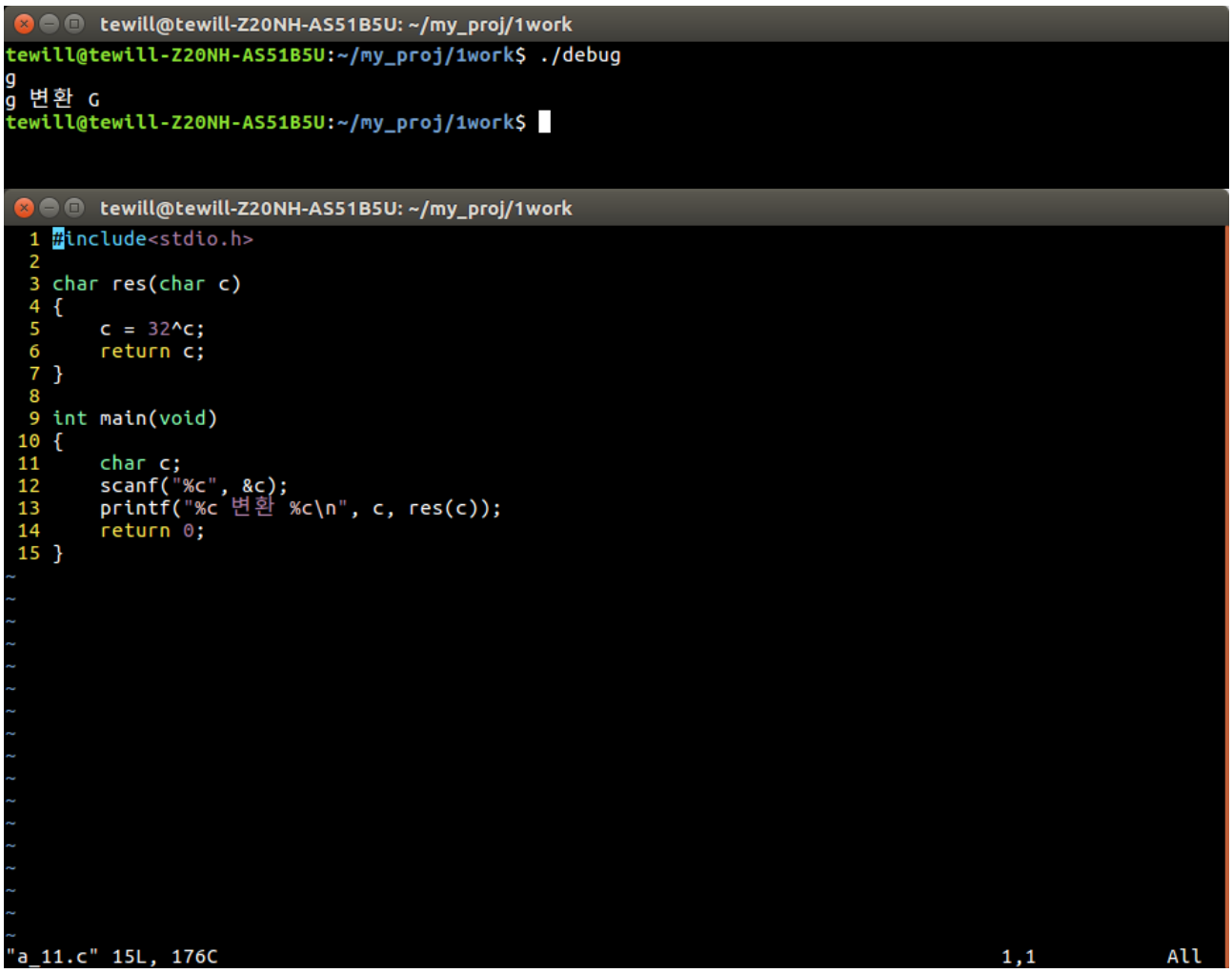
-



21. 아래 설명을 보고 프로그램을 작성한다.
단 한 번의 연산으로 대소문자 전환을 할 수 있는 연산에 대해 기술하시오.

덧셈 혹은 뺄셈 같은 기능이 아님

-



22. 변수의 정의를 기술하시오.

- 메모리 공간 안에서 값을 저장할 수 있도록 한 것이 변수이다. 결국 변수 명은 주소와 같은 말이고, 해당 주소가 나타내는 위치에 들어있는 값이 변수의 값이다.

23. 포인터의 정의를 기술하시오.

- 메모리 공간 안에서 주소를 저장할 수 있도록 한 것이 포인터이다. 변수같은 경우 값을 저장하는 반면, 포인터는 주소를 값으로 저장한다.

24. 함수 포인터의 정의를 기술하시오.

- 해당 함수의 주소를 저장하는 것이 함수 포인터이다. 주소를 통해 함수를 받을 수 있으며, 이를 꺼내 사용할 수 있다. 즉, 함수도 결국 주소와 동일하다.

25. 파이프라인은 언제 깨지는가 ?

- call 등 jmp 기능이 사용될 때 파이프라인이 깨지게 된다.

26. 메모리 계층 구조에 대해 기술하시오.

- 메모리는 cpu 내부에 레지스터, 캐시가 존재하며, 외부에 메모리, 하드 디스크가 존재한다. 레지스터와 캐시는 cpu 내부에 있기 때문에 속도가 빠르지만, 가격이 비싸기 때문에 피라미드 형태로 메모리 계층 구조가 생기게 된다.

27. C 언어의 기본 메모리 구조에 대해 기술하시오.

- 스택과 힙, 데이터와 코드로 이루어져 있다. 스택은 지역변수이며, 힙은 동적으로 할당된 메모리이다. 코드는 함수를 의미한다.

28. 디버깅하면서 보는 변수의 주소는 무엇일까 ?
우리가 사용하는 윈도우나 리눅스, 맥(유닉스)에서 보는 변수 주소는 진짜 주소일까 아닐까 ?

알고 있는대로 기술하시오.

- 가상 메모리를 통해 사용자가 보기 쉽게 보여주는 가짜 주소이다.

29. 아래 설명을 보고 프로그램을 작성하시오.
이름과 급여를 저장하도록 만든다.

이름은 마음대로 고정시키도록 하고 급여는 rand() 를 활용

급여의 평균을 출력하고 가장 높은 한 사람의 이름과 급여를 출력하시오.

(값이 같을 수 있음에 유의해야 한다)

-

tewill@tewill-Z20NH-AS51B5U: ~/my_proj/1work

```
29
30 void mean(user *us, int size)
31 {
32     int i, sum = 0;
33     for(i = 0; i<size; i++)
34     {
35         sum += us[i].pay;
36     }
37     sum /= size;
38
39     printf("급여 평균 : %d\n", sum);
40 }
41
42 void set_data(user *us, int size)
43 {
44     int i;
45
46     for(i = 0; i < size; i++)
47     {
48         scanf("%s", us[i].name);
49         us[i].pay = rand()%30+1;
50     }
51 }
52
                                                    29,0-1        68%
```

tewill@tewill-Z20NH-AS51B5U: ~/my_proj/1work

```
 1 #include <stdio.h>
 2 #include <time.h>
 3 #include <string.h>
 4 #include <stdlib.h>
 5 struct userdata
 6 {
 7     char name[34];
 8     int pay;
 9 };
10
11 typedef struct userdata user;
12
13 void explo(user *us, int size)
14 {
15     int i;
16     user tmp = us[0];
17
18     for(i = 0; i < size; i++)
19     {
20         if(tmp.pay < us[i].pay)
21         {
22             tmp = us[i];
23         }
24     }
25
26     printf("최고 급여자 : %s\n", tmp.name);
27
28 }
29
                                                    1,1           Top
```

tewill@tewill-Z20NH-AS51B5U: ~/my_proj/1work

```
52
53 int main(void)
54 {
55     user data[10];
56     int size;
57     size = sizeof(data)/sizeof(user);
58
59     srand(time(NULL));
60
61     set_data(data, size);
62     mean(data, size);
63     explo(data, size);
64     return 0;
65 }
                                                    65,1          Bot
```

30. 리눅스에서 디버깅을 수행하기 위한 프로그램 컴파일 방법을 기술하시오.

-gdb

31. vi 활용 방법
vi 에서 코드가 정렬이 잘 안되어 있다.

이 상황에서 어떻게 해야 예쁘고 깔끔하게 정렬되는가 ?

- ‘v’ 를 눌러서 원하는 영역까지 방향키로 설정. 그 후 ‘=’ 를 클릭하면 정렬이 된다.

32. 최적화 기법에 대한 문제다.
프로그램을 최적화하는 컴파일 옵션을 적고

반대로 어떠한 최적화도 수행하지 않는 컴파일 옵션을 적도록 한다

- gcc 에서 최적화를 수행하지 않는 명령어는 -0O 이며, 반대의 경우 -5O 이다.

33. gdb 를 사용하는 이유를 기술하시오.

- 디버깅을 하기 위해서.

34. 기계어 레벨에서 스택을 조정하는 명령어는 어떤것들이 있는가 ?
직접 연산을 하는 경우를 제외하고 기술하도록 한다. (4 가지)

-