1. 단 한 번의 연산으로 대소문자 전환을 할 수 있는 연산에 대해 기술하시오
C 언어로 프로그램을 구현하시오.

```
#include <stdio.h>

int main(void)
{
        char a;
        printf("대문자로 변환할 문자를 입력하시오: ");
        scanf("%c", &a)

        printf("%c의 대문자는 %c입니다.\n", a, a-32);

        return 0;
}
```

2.Stack 및 Queue 외에 Tree 라는 자료구조가 있다.
이 중에서 Tree 는 Stack 이나 Queue 와는 다르게 어떠한 이점이 있는가 ?

stack이나 queue와는 다르게 노드를 통해 데이터에 접근 가능하며 단일 입출방식이 아닌 link된 노드에 접근해 중간의 데이터를 조작할 수 있다.
(데이터 조작에 이점이 있음)

3. 임의의 값 x가 있는데, 이를 4096 단위로 정렬하고 싶다면 어떻게 해야할까 ?
C 언어로 프로그래밍 하시오.

```
#include <stdio.h>

int main(void)
{
  int x;
  int nDigit;
  printf("변환할 숫자를 입력하시오:");
  scanf("%d", &x);

 printf("변환된 수:");

  while(x/4096)
  {
        nDigit = x /4096;
        printf("%d", nDigit);
```

```
        x=x%4096;
 }

return 0;
}
```

4. int p[7] 와 int (*p)[7] 가 있다. 이 둘의 차이점에 대해 기술하시오.

- int p[7]은 int형의 값을 7개 저장할 수 있는 공간을 가진 배열
- int (*p)[7]은 int형 데이터 타입의 값을 저장하는 공간의 주소를 7개 저장할 수 있는 공간을 가진 포인터배열

5. 다음을 C 언어로 구현하시오.
이번 문제의 힌트를 제공하자면 함수 포인터와 관련된 문제다.

아래와 같은 행렬을 생각해보자!

1 2 3

1 2 3

sapply(arr, func) 이라는 함수를 만들어서 위의 행렬을 아래와 같이 바꿔보자!
sapply 에 func 함수가 연산을 수행하는 함수로 만들어야 한다.

1 2 3

1 4 9

```
#include <stdio.h>

int squarTwoRow(int *a[3])
{
    int i, j;



    for(j=0; j < 3; j++)
    {
```

```
        a[1][j] = a[1][j]*a[1][j];
    }

 return 0;
}
void sapply(int (*arr)[3], int *func(int *a[3]))
{
            int i, j;

            *func(arr);

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

int main(void)
{
   int array[2][3] = {{1,2,3}, {1, 2, 3}};
   int *func(int *a[3]);
   func = squarTwoRow;
   sapply(array, func);

   return 0;
}
```

6. 다음 사항에 대해 아는대로 기술하시오.

Intel Architecture 와 ARM Architecture 의 차이점은 무엇인가 ?

- x86 은 메모리 에서 메모리로 연산이 가능
- ARM 은 로드/스토어 아키텍처라고 해서 메모리에서 메모리로 연산 불가능

7. 이것이 없으면 C 언어의 함수를 호출할 수 없다.

여기서 이야기하는 이것은 무엇일까 ?

메모리주소

8.아래 내용을 C 로 구현해보도록 하자.
3x^2 + 7x 를 1 ~ 2 까지 정적분하는 프로그램을 구현해보라.
예로 x 에 1 이 들어가면 3x^2 = 9 가 된다.

```
int SecondEqIntegral(float a, float b, float c, int start, int end)
{
        float xe3=0, xe2=0, xe1=0, xs3=0, xs2=0, xs1=0;
        bool bThird=false;
        bool bSecond=false;
        bool bFirst=false;
        float result;


        if(a != 0)
        {
           a = a/3;
           nThird = true;
        }

        if(b != 0)
        {
           b = b/2;
           nSecond = true;
        }

        if(c!= 0)
        {
           bFirst = true;
        }

        if(nThird)
        {
           xe3 = end*end*end;
           xs3 = start*start*start;
        }

        if(nSecond)
```

```
        {
            xe2 = end*end;
            xs2 = start*start;
        }

        if(bFirst)
        {
            xe1 = end;
            xs1 = start;
        }



        result = a(xe3-xe3)+b(xe2-xe2)+c(xe1-xs1);

        return result;
}
```

9. Memory Hierarchy(메모리 계층 구조)에 대해 기술하시오.

순위	속도	용량
1	register	disk
2	cache	momory
3	momory	cache
4	disk	register

10. C 언어에서 중요시하는 메모리 구조에 대해 기술하시오.
(힌트: Stack, Heap, Data, Text 에 대해 기술하시오.)

Stack(지역변수, 일시적 생성 변수 저장)
Heap(동적메모리 할당 메모리저장)
Data(전역변수, 정적변수 저장)
Text(함수, 코드 저장)

11. 파이프라인이 깨지는 경우에 대해 기술하시오.

goto명령어 또는 재귀함수 호출과 같이 assembly의 jmp에 해당하는 명령문을 문함하는 내용은 call을 통해 파이프라인 흐름을 깨트리게 됨

12. void (* signal(int signum, void (* handler)(int)))(int)라는 signal 함수의 프로토타입을 기술하시오.

반환타입: void
함수이름: signal
인자: int signum, void (*handler)(int)

13. goto 를 사용하는 이유에 대해 기술하시오.

예외문 처리 또는 메모리주소 jmp가 필요한 경우 goto를 활용할 수 있음

14. 아래 질문에 적절한 C 코드를 구현하시오.
TI Cortex-R5F Safety MCU is very good to Real-Time System.
위의 문장에서 Safety MCU 가 몇 번째 부터 시작하는지 찾아내보자!
(배열의 인덱스로 표기해도 상관없고, 실제 위치로 표기해도 상관없으며 둘 중 무엇인지 표기하시오)

15. 아래 질문에 대한 C 코드를 작성하시오.
배열에 아래와 같은 정보들이 들어있다.

2400, 5000, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 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, 978, 456, 234756 , 234 ,4564 ,3243, 876,645, 534, 423, 312, 756, 235 ,75678

여기서 가장 빈도수가 높은 3개의 숫자를 찾아 출력하시오!

```
#include <stdio.h>
int array[] ={

2400, 2400, 2400, 2400, 2400, 2400, 2400, 2400, 2400, 2400, 2400, 2400,
```

```
2400, 2400, 2400, 2400, 2400, 2400, 2400, 2400, 2400, 2400, 2400, 2400, 2400,
2400, 2400, 5000, 5000, 5000, 5000, 5000, 5000, 5000, 5000, 5000, 5000, 5000,
5000, 5000, 5000, 5000, 5000, 5000, 5000, 5000, 5000, 5000, 5000, 5000, 5000,
5000, 5000, 5000, 5000, 5000, 5000, 5000, 5000, 5000, 5000, 5000, 5000, 500, 500,
500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500,
500, 500, 500, 500, 500, 500, 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, 978, 456, 234756 , 234 ,4564 ,3243, 876,645, 534, 423, 312, 756, 235
,75678

}

int main(void)
{
          int nEnd=0;
          int nNumArray[] = {0};
          int nIndex[] = {0};
          int i, j, temp=0;

         nEnd = sizeof(array)/sizeof(int);

         for(i=0; i < nEnd; i++)
         {
           nNumArray[array[i]]++;
         }

         for(i=0; i < nEnd; I++)
         {

           for(j=i; j < nEnd; j++)
           {
                 if(nNumArray[i] < nNumArray[j])
                 {
                           temp = nNumArray[i];
                           nNumArray[i] = nNumArray[j];
                           nNumArray[j] = temp;

                           nIndex[i] = j;
                  }
```

```
            }
          }

        for(i=0; I < 3; I++)
        {
           printf("%d번째 많이 나타나는 수 %d, 회수 %d\n",nIndex[i],
nNumArray[i]);
        }

         return 0;
}
```

16. 아래 질문에 적절한 C 코드를 구현해보자.
내용
16 비트 ADC 를 가지고 있는 장치가 있다.
보드는 12 V 로 동작하고 있고 ADC 는 -5 ~ 5 V 로 동작한다.
ADC 에서 읽은 값이 12677 일 때 이 신호를 전압으로 변환해보자!
(실제 보드가 없으므로 단순히 C 로 구현하면 된다)

```
float ChangeDigitalToAnalog(int nDigitalVal)
{
    float fVolt;
    int nDiv;
    nDiv = (pow(2,16)-1)>>1;


  if(fVolt < nDiv)
  {
     fVolt = -nDigitalVal/(nDiv)*5;
  }
  else
  {
     fVolt = nDigitalVal/(nDiv)*5;
  }

  return fVolt;
}
```

```
int main(void)
{
   float fVolt;
   fVolt = ChangeDigitalToAnalog(12677);
   printf("volage변환값: %f\n", fVolt);

   return 0;
}
```

17. 설명을 읽고 C 코드를 구현해보자.
24 비트 DAC 장치가 있다.
이 장치는 -12 V ~ 12 V 로 동작하며 보드는 5 V 로 동작한다.
DAC 에서 나온 전압이 9.7 V 일 때 어떤 디지털 신호를 입력 받은것인지 파악해보자!

```
int ChangeAnalogToDigital(float fVolt)
{
    int nDigit;
    int nDiv=25;

    nDigit = (int)fVol/(nDiv)*((pow(2, 24)-1) >> 1);

    return nDigit ;
}

int main(void)
{
   int nDigit;
   nDigit = ChangeAnalogToDigital(9.7);
   printf("Digital 변환값: %d\n", nDigit );

   return 0;
}
```

18.다음 질문에 적절한 답을 적으시오(주관식)
운영체제의 5 대 요소 5 가지를 적으시오.

19. 아래 자료 구조를 C 언어로 구현해보시오.
Stack 자료구조를 아래와 같은 포맷에 맞춰 구현해보시오.

(힌트: 이중 포인터)
ex)

```
typedef struct _stack{
int data;
struct _stack *next;
}stack;


int main(void)
{

    stack *top = NULL;

    push(&top, 1);

    push(&top, 2);

    printf("data = %d\n", pop(&top));

}

void push(stack *top, int data)
{
  stack *temp;
  temp = (stack *)malloc(sizeof(stack));

  temp->data = data;
  top = temp;
  free(temp);
}

int pop(stack *top)
{
  int data;
```

```
  stack *temp;

  data = top->data;

  temp = top;

  top = temp->next;

  free(temp):

  return data;

}
```

20. 다음 질문에 답하시오.
Binary Tree 나 AVL Tree, Red-Black Tree와 같이 Tree 계열의 자료구조를 재귀 호출 없이 구현하고자 한다.
이 경우 반드시 필요한 것은 무엇인가 ?

21. 다음 자료 구조를 C 로 구현하시오.
Binary Tree 를 구현하시오.
초기 데이터를 입력 받은 이후 다음 값이 들어갈 때 작으면 왼쪽 크면 오른쪽으로 보내는 방식으로 구현하시오.
삭제 구현이 가능하다면 삭제도 구현하시오.

22. 다음 질문에 대해 기술하시오.
AVL 트리는 검색 속도가 빠르기로 유명하다.
Red-Black 트리도 검색 속도가 빠르지만 AVL 트리보다 느리다.
그런데 어째서 SNS 솔루션등에서는 AVL 트리가 아닌 Red-Black 트리를 사용할까 ?

23. 다음 자료구조를 C 로 구현하시오.

AVL 트리를 C 언어로 구현하시오.
AVL 트리는 밸런스 트리로 데이터가 1, 2, 3, 4, 5, 6, 7, 8, 9 와 같이
순서대로 쌓이는 것을 방지하기 위해 만들어진 자료구조다.

24. 다음 질문에 답하시오(주관식)
리눅스 실행 파일 포맷이 무엇인지 적으시오.

디렉토리, 파일, 바이너리

25. 다음 질문에 답하시오.
리눅스 운영체제에서 프로세스와 스레드를 구별하는 방법에 대해 기술하시오.

($) ps -aux 실행시 //프로세스는 PID값 가짐 ($) ps -axms //스레드는 PID값이 없음

26. Context Switching 에 대해 기술하시오.
Multi-Tasking 의 핵심에 해당하는 기술인 Context Switching 에 대해 기술하시오.

메모리에 로드되어 실행되고 있는 프로세스가 바뀔 때 이를 context switching이라고 함

27. O(N) 과 O(1) 알고리즘에 대해 고려해보자.
둘 중 무엇이 더 좋은것인가 ?
이유를 기술하시오.

O(N)이 알고리즘 처리 속도에서 O(1)보다 더 좋다. O(N)은 시간이 지남에 따라 처리속도가 정비례로 증가함

28. 가상 메모리란 무엇인지에 대해 기술하시오.
리눅스 커널의 task_struct 구조체를 살펴보면 mm_struct 라는 구조체가 있다.
이 구조체의 관점에서 기술하시오.

29. MMU 에 대해 기술하시오.
Cortex-M, Cortex-R, Cortex-A 등의 다양한 ARM 아키텍처가 존재한다.
이 중 Cortex-A 에는 반드시 MMU 가 탑재되는데 MMU 가 하는 일에 대해 자세히 기술하시

오.

프로세서의 메모리관리를 담당하며 가상메모리 주소와 물리 메모리 주소의 변환 및 관리를 수행

30. 소프트웨어 인터럽트에 대해 생각해보자.
리눅스 시스템에서 소프트웨어 인터럽트 번호는 0x80 에 해당한다.
소프트웨어 인터럽트가 무엇인지 기술하고 이것의 동작 메커니즘에 대해 기술하시오.

인터럽트 벡터 호출 -> 인터럽트 핸들러 호출 -> 인터럽트 핸들 수행 -> 원래 코드로 복귀

31. 운영체제 메모리 관리 기법중 페이징에 대해 기술하시오.
리눅스 커널의 task_struct, mm_struct, vm_area_struct 를 기준으로 적어도 무방하며
CP15 레지스터등의 내용등 자유롭게 기술해도 무방하다.
(힌트: 32 비트 기준 10, 10, 12)

데이터를 저장하기 위해 보통 32비트 기준 4KB단위의 페이지를 규격으로 메모리 저장공간을 할당 및 관리

32. 다음 디바이스 드라이버를 리눅스 상에서 구현해보시오.
Character Device Driver를 아래와 같이 동작하게 만드시오.
read(fd, 1, 10)을 동작시킬 경우 2번째 인자 ~ 3번째 인자 까지의 덧셈을 반환하도록 한다.
write(fd, 1, 5)를 동작시킬 경우 2번째 인자 ~ 5번째 인자 까지의 곱셈을 반환하도록 한다.
lseek(fd, 7, SEEK_END)를 동작시킬 경우 2번째 인자값을 반환하고 Kernel 내에서 "SEEK_END means end"를 출력하게 하라!
close(fd)를 수행하면 Kernel 내에서 "Finalize Device Driver"가 출력되게 하라!

33. 이번 멜트 다운 버그와 관련한 문제다.
OoO(Out-of-Order)인 비순차 실행에 대해 기술하라.
멜트 다운 버그가 이 OoO 기능과 관련이 있다.
그러나 해당 내용까지 기술할 필요는 없다.

cpu스케줄링과 관련된 사항으로 개발자의 의도와는 다르게 cpu가 비순차적으로 스케줄링되어 작동함

34. "Hello World" 가 출력되도록 인라인 ARM 어셈블리를 활용해보자.
아래와 같은 형식을 사용해보도록 하라.

```
#include <stdio.h>

char msg[15] = "Hello World";

int main(void)

{

        register unsigned char *r5 asm("r5") = NULL;

        // 여기 한줄만 채우면됨

        printf("r5 = %s\n", r5);

        return 0;

}
```

35. 어셈블리 프로그래밍을 해보자.
아래 형식을 기반으로 배열의 내용을 "Hello Test" 로 출력되도록 어셈블리 프로그래밍 하시오.
원래 배열에는 "Hello World" 가 들어있다.

```
#include

char msg[15] = "Hello World";

int main(void)

{

        register unsigned char *r5 asm("r5") = NULL;

        register unsigned char r6 asm("r6") = 0;

        r5 = msg;

        printf("r5 = %s\n", r5);
```

```
        // asm volatile 부분에 어셈블리 명령어를 채워서 기능이 동작하게 만들면됨
        asm volatile("mov r6, #0x54\n"
                     "strb r6, [r5, #6]\n");
        printf("msg = %s\n", msg);
        return 0;
}
```

36. 리눅스 시스템 프로그래밍 문제로 System Call 을 활용하도록 하라.
디렉토리 내에 들어 있는 모든 File 들을 출력하는 Program을 작성하시오.

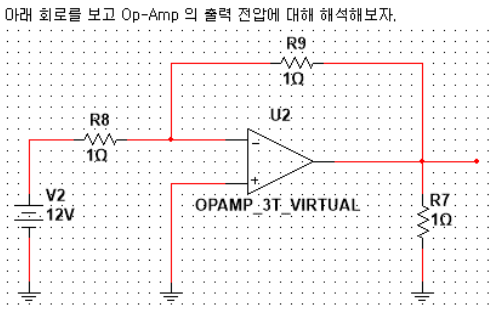
37. 리눅스 시스템 프로그래밍 기법을 활용하여 369 게임을 프로그래밍 해보시오.
2 초 내에 값을 입력하게 하시오.
박수를 쳐야 하는 경우를 Ctrl + C를 누르도록 한다.
2초내에 값을 입력하지 못할 경우 게임이 오버되게 한다.
Ctrl + C를 누르면 "Clap!"이라는 문자열이 출력되게 한다.

38. 리눅스 시스템 프로그래밍 문제다.
랜덤한 이름을 가지도록 디렉토리 3개를 만들고
각각의 디렉토리에 5 ~ 10개 사이의 랜덤한 이름(길이도 랜덤)을 가지도록 파일을 만들어보자!
모든 디렉토리를 순회하며 3 개의 디렉토리와 그 안의 모든 파일들의 이름 중
a, b, c 가 1개라도 들어있다면 이들을 출력하라!
출력할 때 디렉토리인지 파일인지 여부를 판별하도록 하시오.

39. 시스템 콜 관련 문제다.

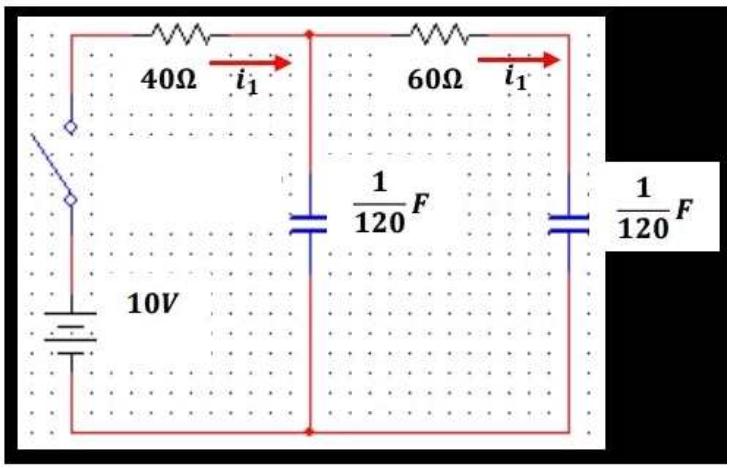
ARM 에서 System Call 을 사용할 때 사용하는 레지스터를 적으시오.

40. 다음 Op-Amp 를 해석해보자.

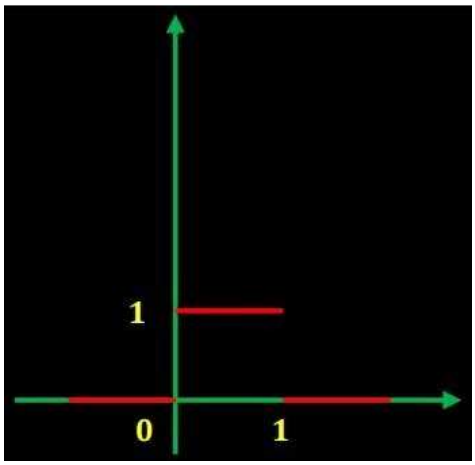


Vout = 12*1/2 = 6V

41. 아래 회로의 동작 함수에 대해 분석해보자.
아래 회로를 해석하기 위해 라플라스 변환을 활용해보길 바란다.



42. 다음 그림을 보고 문제를 푸시오.
아래 그림의 신호에 대한 푸리에 적분을 수행해보시오.



43. 다음 연립 방정식을 푸시오.
(힌트: 크래머 공식, 역 행렬 방식, determinant 등을 활용하면 쉽게 풀 수 있다)

$$3x + 6y + 9z = 60$$
$$4x + 2y + 6z = 40$$
$$3x + 2y + z = 20$$

44. 아래 그림을 보고 미분 방정식을 푸시오.
전형적인 2 계 비동차 미분 방정식 문제에 해당한다.

$$3y'' + 4y' + y = x^2 + 4$$

45. 다음 행렬의 역행렬을 구하시오.
힌트: 가장 편하게 푸는 방법은 determinant 를 활용하는 것이다.

$$\begin{pmatrix} 1 & -2 & 0 \\ 3 & 1 & 5 \\ -1 & 2 & 3 \end{pmatrix}$$

46. 레이더 범위 방정식에 대한 문제다.
여기서 도출할 수 있는 핵심적 결론은 무엇인지 기술하시오.

$$R_{max} = \left(\frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 k T_o B F L (SNR)} \right)^{\frac{1}{4}}$$

$$SNR = \frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 k T_o B F L R^4}$$

47. 공중의 방송국이라 불리는 EC-130H 에 대한 문제다.
미국과 전쟁을 하게되면 이 기체가 떠서 적국의 모든 방송국을 방송 불가능하게 재밍을 한다.
그리고 주파수를 전부 이 기체가 스니핑하여 어떤 방송에서도 이 기체가 하는 방송만을 듣게 만들 수 있다.
실제 미군이 아랍어 방송을 한 사례로 매우 유명하다.
아래 사진의 EC-130H 기체가 이와 같은 기술을 사용하는데 필요한 핵심 기술은 무엇인가 ?
(힌트: 영어 단어 3 개)

ECM(Electronic Counter Measure)

48. 연립 미분 방정식을 풀어보자.
힌트: 라플라스 변환

$$x'' - 3x' + 4y' + y = 4$$
$$2y' - x' + 3y = 0$$
$$x(0) = x'(0) = y(0) = 0$$

49. 벡터 미적분학 문제다.
이 벡터 함수에 대한 Gradient, Divergence, Curl 을 구하시오.

$$\{x^2 + 2y + z\ln(x)\}\hat{\imath} + \{y^2 z + e^z\}\hat{\jmath} + \{xyz + 2x^2 y e^z + z^3\}\hat{k}$$

50. 가우스-조르단 소거법을 C 로 구현하시오.
연립 방정식 문제를 C 가 풀 수 있게 구현하면 된다.
이것은 주로 로봇 팔의 제어등을 수행하는데 활용되는 기법이다.

51. 벡터의 사칙 연산, 내적 및 외적을 C 로 구현하시오.
힌트: 벡터의 외적은 3 by 3 행렬의 determinant 를 구하는것과 유사하다.

52. 역행렬을 구하는 프로그램을 구현하시오.
정밀 제어를 수행하기 위해 칼만 필터등을 구현하는데도 이 기능을 필수적이다.

53. 펌웨어 문제다.
간단한 LED 를 제어하고자 한다.
이때 사용하는 Peripheral 을 무엇이라고 하는가 ?

GPIO(General Purpose Input Output)

54. 디지털 신호 처리를 하는 이유에 대해 기술하시오.
힌트: 수치 해석, 샘플링 타이밍

자연상에 존재하는 아날로그 신호를 수치해석과 디지털 샘플링을 통해 디지털화해 분석 또는 응용가능한 데이터로 만들기 위해 사용

55. 푸리에 변환을 통해 얻는 정보는 무엇인가 ?
해당 정보를 어디에 활용하는 것인지 기술하시오.

시간도메인에서 발생 또는 존재하는 데이터를 주파수 도메인으로 변환하여 데이터의 발생 빈도와 같은 특성을 분석할 수 있음

56. 라플라스 변환을 왜 배워야 하는가 ?
본인이 아는대로 기술하시오.

57. 4 Point DFT 를 C 로 구현하시오.
실제 DFT 는 연산량이 많아서 고속 신호 처리에는 적합하지 않아 FFT 알고리즘을 만들게 된다.

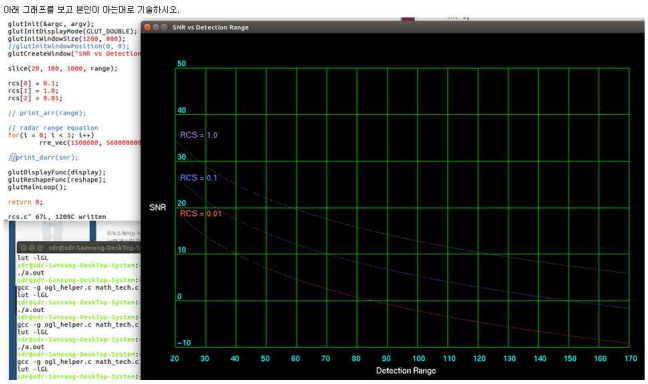
58. 8 Point FFT 는 C 로 구현해보시오.
힌트: Reverse Order 와 Butterfly 연산에 유의하라.

59. 다음 Peripheral 들의 기능에 대해 기술하시오.
PWM, I2C, SPI, UART, ADC

- PWM: Pulse의 주기(발생 width)를 조정해 스위칭 또는 전압인가의 주기를 조정
- I2C: master-slave간 통신하는 방식으로 2 line을 통해 master-slave가 통신
- SPI: mster-slave간 통신하는 방식으로 MOSI(Master Output Slave In)핀을 통해 통신
- UART: 비동기 직렬통신 프로토콜로 컨트롤러 간 비동기 통신을 가능하게 함
- ADC: 아날로그 신호를 디지털 신호로 변경하는 부분으로 아날로그 신호를 디지털화

60. RL 회로에 대한 해석을 수행해보시오.
처음 전압을 인가하여 자기장을 에너지로 충전할 때와
이후 충전한 자기장을 에너지로 배출할 때에 대한 해석을 수행하시오.
추가적으로 역학에는 관성 개념 존재한다(버스가 급 정지 한다던지, 정지한 차가 급 출발 한다던지)
전기 분야에서 관성에 해당하는 내용은 무엇인지 기술하시오.
(힌트: RL 회로에서 이 전기적 관성을 살펴볼 수 있다)

61. RCS 란 무엇인가 ?



62. 레이더는 즉각적으로 적의 움직임을 파악할 수 있다.
레이더는 전자기파를 방출한다.
전자기파의 속도가 얼마나 빠르길래 적이 움직이자마자 속도값과 거리값을 알아채고 방향까지

알 수 있는것일까 ?
(힌트: 맥스웰이 맥스웰 방정식을 발표하고 이것을 예언했고 이로써 도출되는 결론과 관계된다)

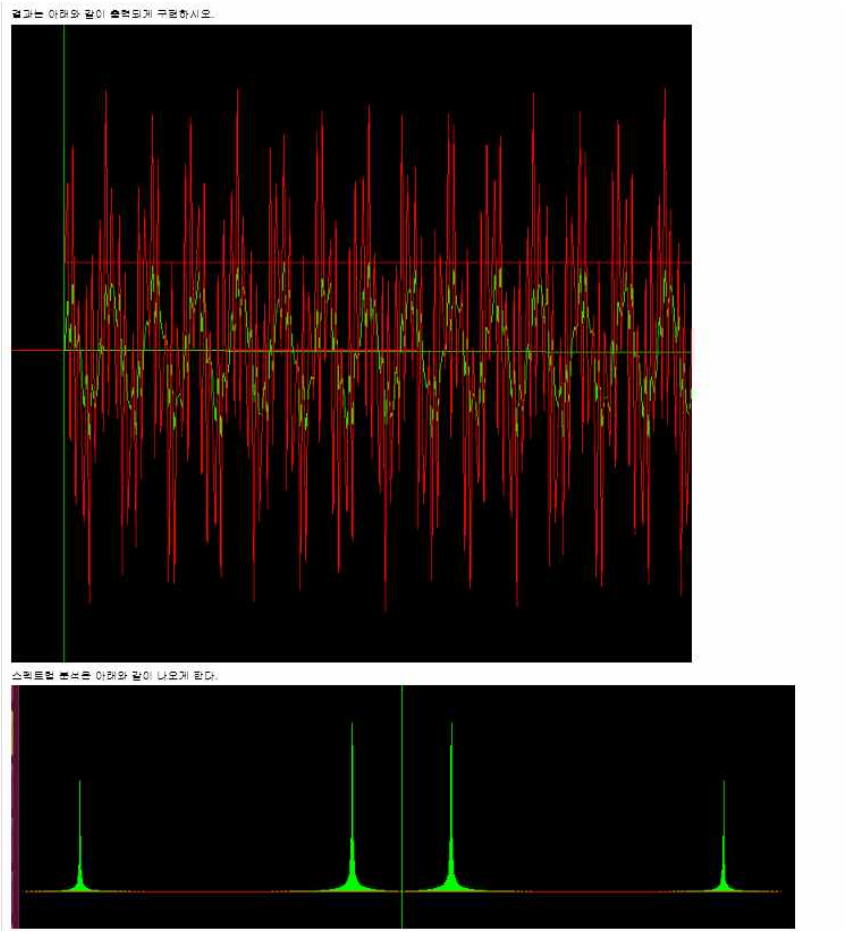
63. VHDL 혹은 Verilog 를 사용하여 주파수 분할기를 구현해보시오.
주파수 분할기를 FPGA 를 활용하여 PWM 제어를 수행하는 작업을 수행할 때 매우 중요한 부분이다.
(Vivado HLS 를 사용하는 경우라면 HLS 를 사용해도 무방하다)

64. FPGA 보드에 Petalinux 를 올리고 디바이스 드라이버를 만들어보자.
간단하게 브레드 보드에 LED 를 연결하고 이것을 FPGA 에서 디바이스 드라이버를 통해 제어해보라.

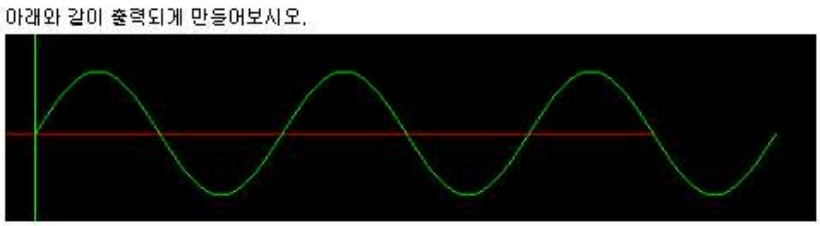
65. FPGA 보드에 Petalinux 를 올리고 PWM 제어를 수행해본다.
이를 통해서 간단한 서보 모터를 구동시켜보도록 한다.

66. FPGA 보드에 petalinux 를 올리고 I2C 제어를 수행해보자.
학원에서 사용하는 대표적 I2C 장비는 Lidar 에 해당한다.
이 Lidar 를 I2C 제어를 통해 제어해보도록 하라.

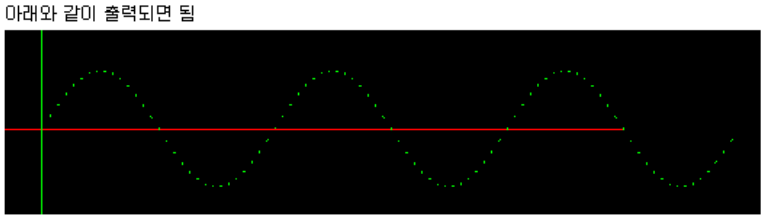
67. 간단한 RC 필터를 C 로 구현하시오.



68. 간단한 sin 파형을 C 언어와 OpenGL 을 활용하여 그려보시오.

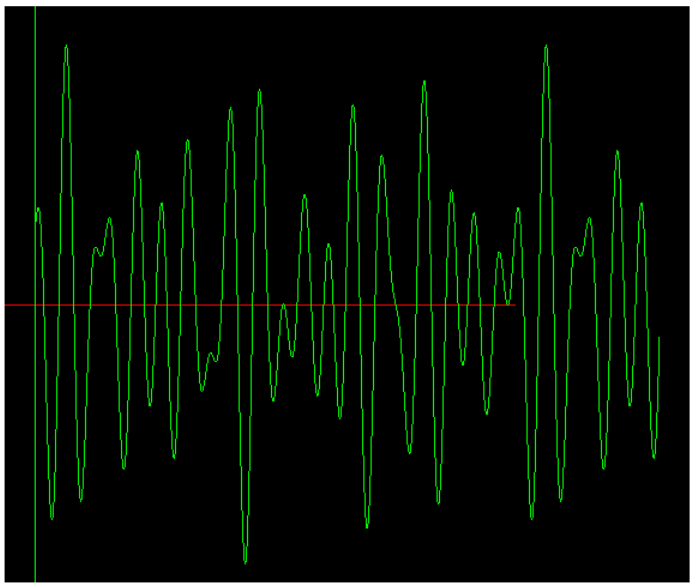


69. sine 파형을 점선으로 찍어보시오.



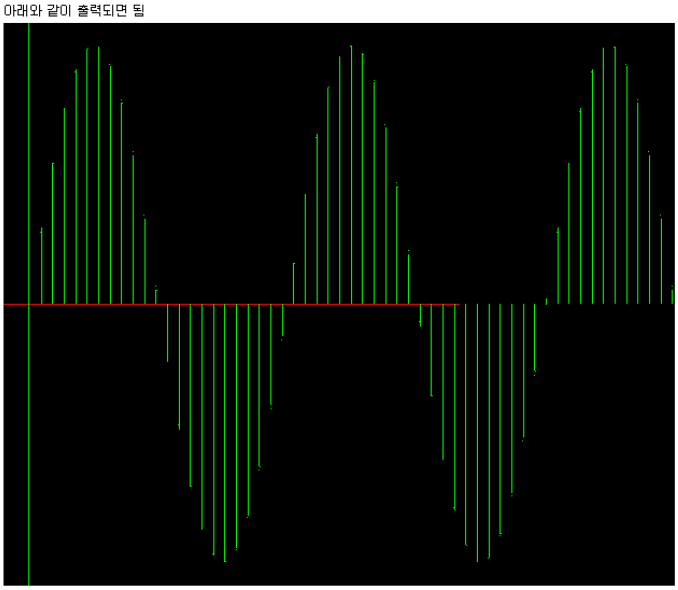
70. sine 파형을 합성하시오.

아래와 같은 합성 파형을 만들어보시오.

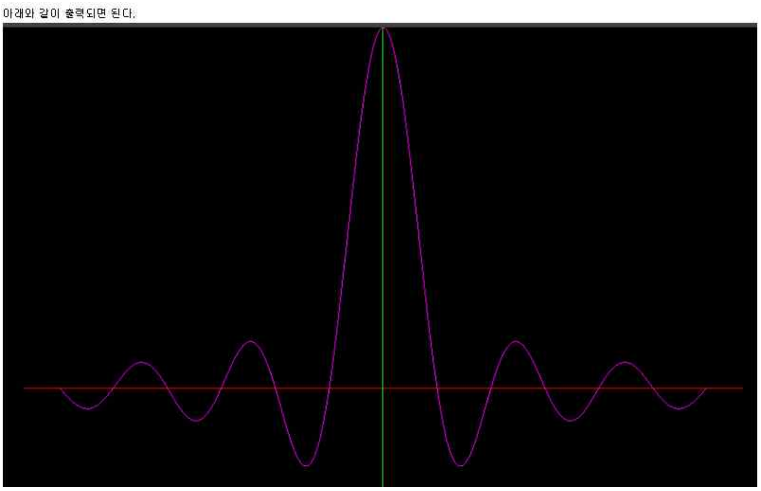


71, 이산 신호 처리를 수행할 수 있도록 신호를 디지털화 하시오.

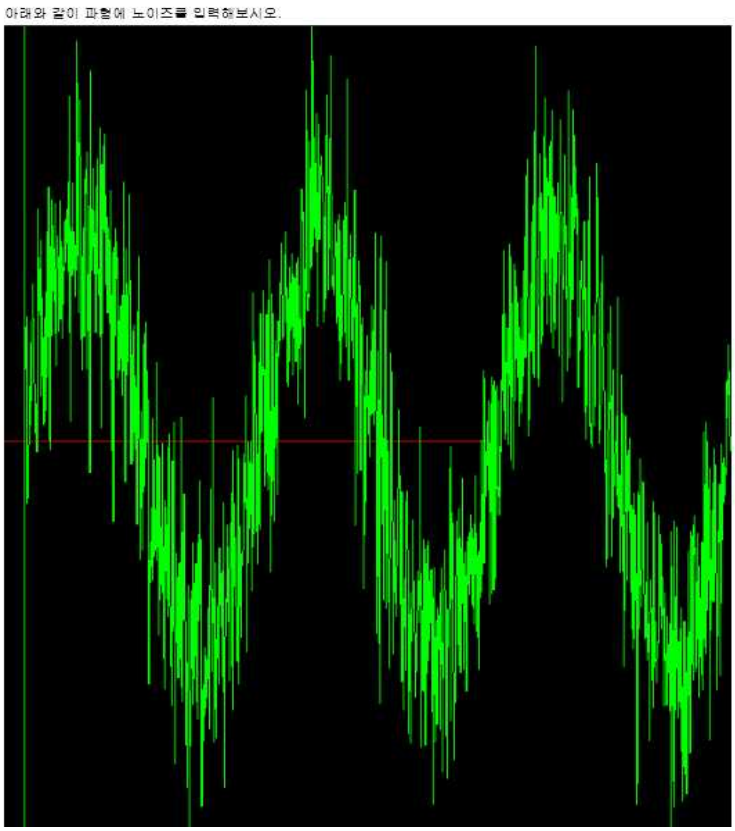
아래와 같이 출력되면 됨



72. 수능에 자주 등장하는 sinc 함수를 구현해보자.

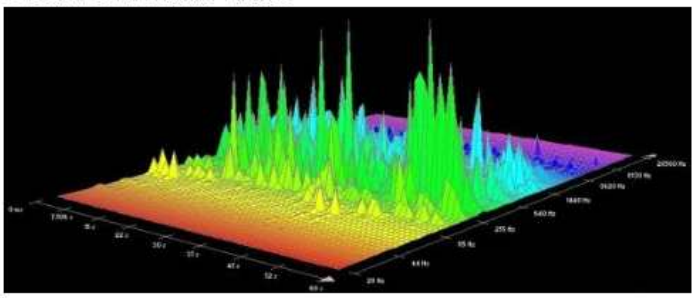


73. 가우시안 노이즈를 구현해보자.



74. 스펙트로 그래프를 구현하시오.

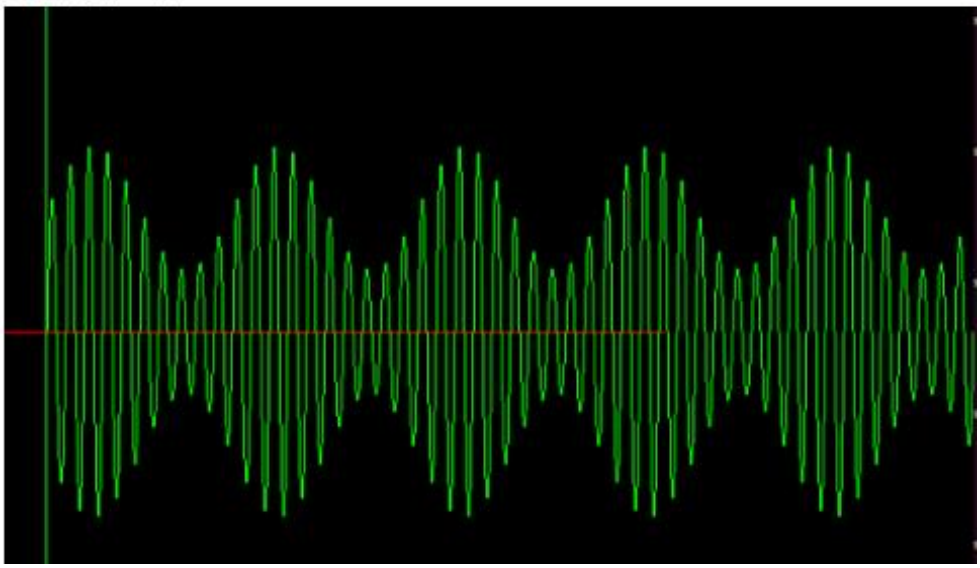
임의의 신호 함수를 만들고 아래와 같이 출력되게 구현하시오.
스펙트럼, 시간, 주파수의 3D로 구성해야함



75. 네비어 스톡스 방정식에 대해 아는대로 기술하시오.
네비어 스톡스 방정식은 유체 역학 뿐만 아니라 공기 역학에서도 매우 중요한 해석 모델이다

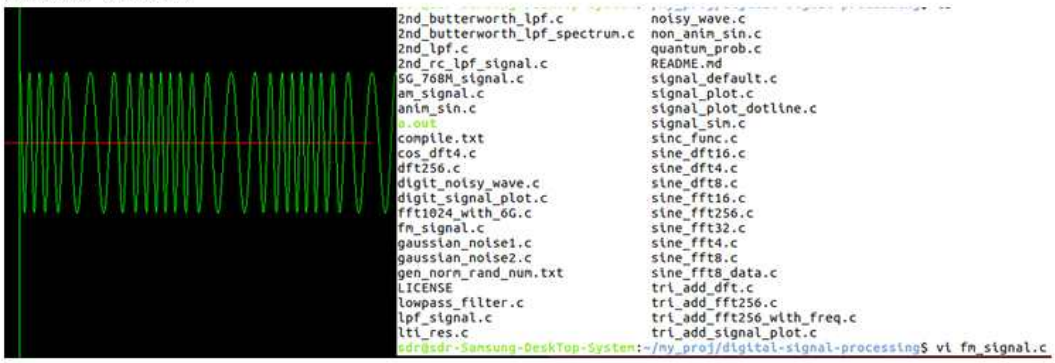
76. AM 변조를 수행하시오.
아래와 같이 출력되면 됨

아래와 같이 출력되면 됨



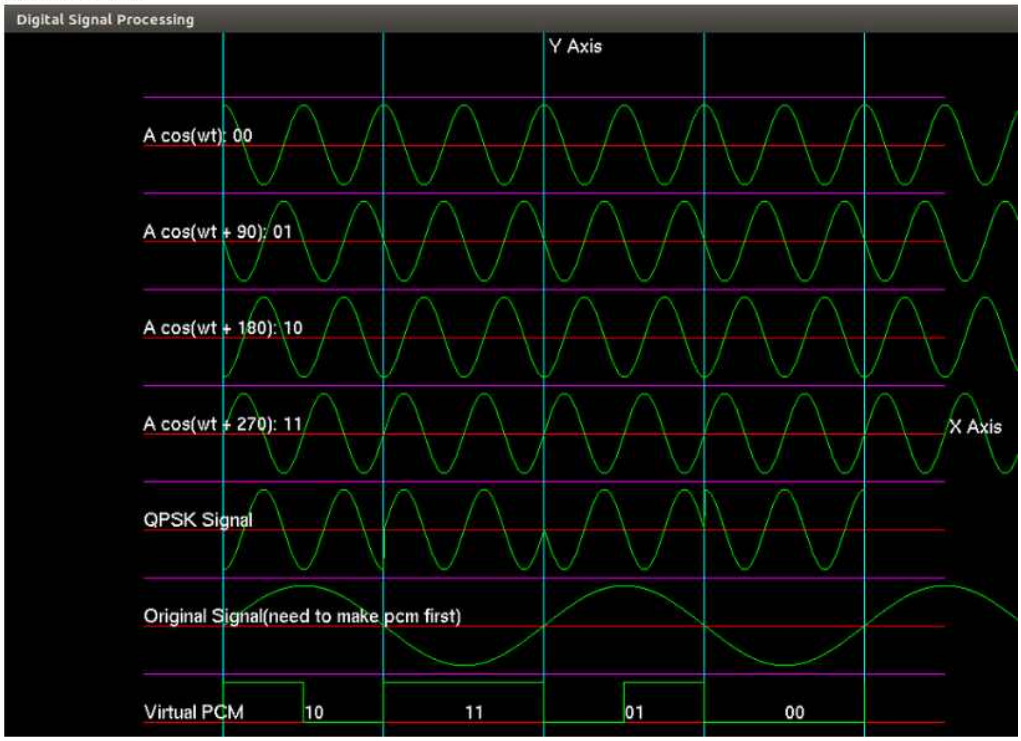
77. FM 변조를 수행해보시오.

아래와 같이 출력되면 됨



78. QPSK 신호를 만들어보시오.

아래와 같이 출력되면 됨



79. 쿼드콥터에 대한 운동 방정식을 기술해보자.

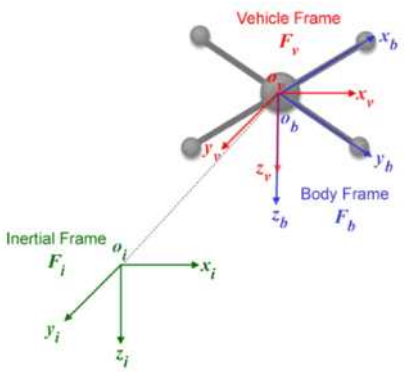
아래와 같이 관성 좌표계, 몸체 좌표계, 기체 좌표계에 입각하여 해석을 하시오.

Reference Frames

운동학 방정식과 Quadrotor 의 동역학에 들어가기 전에 채택된 좌표계와 Reference Frame 을 지정하고
다른 좌표계 간의 변환이 수행되는 방법을 지정하도록 합니다.

다른 좌표 프레임의 사용은 6 DOF(6 개의 자유도)에서 Quadrotor 의 위치와 고도를 식별하는데 필수적입니다.
예로 운동 방정식을 평가하기 위해 quadrotor 에 부착된 좌표계가 필요합니다.
그러나 Quadrotor 에 작용하는 힘과 모멘트는 IMU 센서 값과 함께 Body Frame 을 기준으로 평가됩니다.
마지막으로 Quadrotor 의 위치와 속도는 조종기의 관성 프레임에 대해 GPS 측정을 사용하여 평가됩니다.

그러므로 아래와 같이 3 가지 주요 Reference Frame 이 채택된다:



관성 프레임 $F_i = (\overrightarrow{x_i}, \overrightarrow{y_i}, \overrightarrow{z_i})$ 는
조종기를 원점으로 잡고 있는 지구의 고정 좌표계입니다.
규칙에 따라 x 축은 북쪽을 향하고 y 축은 동쪽을 향하며
z 축은 지구의 중심을 향합니다.

Body Frame $F_b = (\overrightarrow{x_b}, \overrightarrow{y_b}, \overrightarrow{z_b})$ 은
원점이 Quadrotor 의 무게중심(COG) 에 존재합니다.
그리고 축은 Quadrotor 구조와 일직선으로 맞춰져서
x 축의 $\overrightarrow{x_b}$ 는 앞 부분의 motor 방향으로 뻗어나가며
y 축의 $\overrightarrow{y_b}$ 는 $\overrightarrow{x_b}$ 의 우측 motor 방향으로 뻗어나가고
z 축은 $\overrightarrow{z_b} = \overrightarrow{x_b} \times \overrightarrow{y_b}$ 과 같이 외적으로 나타납니다.

Vehicle Frame $F_v = (\overrightarrow{x_v}, \overrightarrow{y_v}, \overrightarrow{z_v})$ 은 관성 프레임이며
원점이 Quadrotor 의 무게중심(COG) 에 존재합니다.
Vehicle Frame 은 F_θ 및 F_ϕ 라는 2 개의 변형을 가집니다.
F_ϕ 은 x_v 와 y_v 가 각각 x_b 와 y_b 에 일직선으로 맞춰지도록
z 축의 z_v 를 중심으로 각도 v 만큼 회전된
Vehicle Frame F_v 를 의미합니다.
F_θ 은 Frame F_ϕ 가 y 축을 중심으로
y_ϕ 만큼 피칭 각도 θ 만큼 회전하여
x_ϕ 와 z_ϕ 가 각각 x_b 와 z_b 에 일직선이 되도록 합니다.

80. 아래와 같이 쿼드콥터를 만들려고 한다.

81. 맥스웰 방정식을 파헤쳐보자.
맥스웰 방정식은 무선 시스템에 있어 매우 중요한 방정식이다.
본인이 알고 있는대로 4 개의 방정식에 대해 기술해보고
Tensor 해석이 가능하다면 2 개의 방정식으로 압축하여 기술해보시오.

82. 현대 전자전에서 핵심인 ECM 에 대해 기술하시오.
(힌트: 채프나 플래어, 재밍 등등중 아무거나 하나 골라 잡고 기술하시오)

83. Larmor Formula 에 대해 기술하시오.
라모 공식이라고도 불리는 Larmor Formula 에 대해 기술하시오.

84. 창 함수에 대해 기술하시오.
창 함수에는 Rectangular, Hamming, Hanning, Kaiser 등이 존재하는데 왜 사용하는 것인가 ?

85. 기본 for 문 활용 문제다.
1 ~ 100 까지의 숫자중 홀수만 더해서 출력해보시오.

```
#include <stdio.h>

int main(void)
{
    int i=0;
    int oddSum=0;

    for(i=1; I <101; I++)
    {
        if(i%2)
        {
            oddSum +=i;
        }
    }

    printf("1~100까지 숫자 중 홀수의 총합: %d\n", oddSum);

    return 0;
}
```

86. 기본 배열 문제다.
1 ~ 100 까지 숫자를 모두 더해서 첫 번째 배열에 저장하고
1 ~ 100 까지 숫자중 홀수만 더해서 두 번째 배열에 저장하고
1 ~ 100 까지 숫자중 짝수만 더해서 세 번째 배열에 저장한다.
다음으로 1 ~ 100 까지 숫자중 3 의 배수만 더해서 네 번째 배열에 저장한다.
각 배열의 원소를 모두 더해서 결과값을 출력하시오.

```
#include <stdio.h>

int main(void)
{
  int array[4];
  int i;

  for(i=1; I < 101; I++)
  {

     if(i%2)
     {
          array[1] += i;
     }
     else
     {
           array[2] += i;
     }

     if((i%3)==0)
     {
           array[3] += i;
     }
   }

   array[0] = array[1] + array[2];

   printf("1~100까지의 합:%d", array[0]);
   printf("1~100까지의 홀수합:%d", array[1]);
   printf("1~100까지의 짝수합:%d", array[2]);
   printf("1~100까지의 3의 배수합:%d", array[3]);

  return 0;
}
```

87. 자기장에 대한 문제다.
비오-사바르 법칙에 대해 기술하시오.

88. 전자기파와 빛의 관계에 대해 기술하시오.
힌트: 이 특성 때문에 초음파등의 음파가 아닌 레이더를 사용하는 것이다.

89. 자유낙하 문제다.
물체가 자유낙하 할 때 종단 속도를 구하기 위한 방정식을 설계해보라.
(힌트: 공기 저항은 속도에 비례한다)

90. 에너지 보존 법칙에 대해 기술하시오.
힌트: 위치 에너지, 병진 운동 에너지, 회전 운동 에너지 관점에서 기술하시오.

91. 차량과 토크의 관계
실제로 차량을 개발하고자할 때 토크 선정이 매우 중요한데 그 이유가 무엇인지 상세히 기술하시오.

92. ADAS 에서 사용하는 다양한 센서를 기술하시오.
힌트: 각 센서들을 왜 이렇게 복합적으로 사용하는 것인가 ?
각각의 이유에 대해 상세히 기술하시오.