최대성

1. 이것이 없으면 사실상 **C** 언어를 사용할 수 없다**.**
C 언어에서 함수를 호출하기 위해서도 이것은 반드시 필요하다**.**
이와 같은 이유로 운영체제의 부팅 코드에서도
이것을 설정하는 작업이 진행되는데 이것은 무엇일까 **?**
답**:** 함수 포인터

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

답**:** 다중 반복문을 한번에 빠져나와야 하는 경우와 같이 **flag** 를 여러번 사용하기 보다는
goto 를 사용하면 빠르고 깔끔하게 나올 수 있다**.**

5. 포인터 크기에 대해 알고 있는대로 기술하시오**.**

답**:** 포인터의 크기는 주소값을 저장하는 공간의 크기를 의미하며 포인터가 가르키는 변수의 크기와는 상관이없고
운영체제에 따라서 포인터의 크기값이 달라진다**.**
예를들면 **32bit** 운영체제인 경우 **4byte, 64bit** 인 경우 **8byte** 의 크기로 할당된다**.**

6. TI Cortex-R5F Safety MCU is very good to Real-Time System.

위의 문장에서 **Safety MCU** 가 몇 번째 부터
시작하는지 찾아내도록 프로그래밍 해보자**.**
(이정도는 가볍게 해야 파싱 같은것도 쉽게 할 수 있다**)**

```
#include <stdio.h>
#include <string.h>
int findStrIndex(char* string, char* findStr){
   int stringLen = strlen(string);
   int findStrLen = strlen(findStr);
   int flag = 1;
   for(int i = 0 ; i< stringLen - findStrLen; i++){
      flag = 1;
      for(int j = 0; j < findStrLen ; j++){
         if(string[i + j] != findStr[j]){
            flag = 0;
            break;
         }
      }
      if(flag == 1)
```

```
            return i;   //index 반환
        }
        return -1; //실패
}
int main(){
    char string[] = "TI Cortex-R5F Safety MCU is very good to Real-Time System.";
    char findStr[] = "Safety MCU";
    printf("%d\n",findStrIndex(string, findStr));
    return 0;
}
```

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

```
#include <stdio.h>
int get_3x3(int N[3][3]){
    int i, j, n;
    printf("3by3 matrix input:\n");
    for(i = 0; i < 3; i++){
        for(j = 0; j < 3; j++){
            printf("%d 행 %d 열 입력:", i, j);
            scanf("%d", &n);
            N[i][j] = n;
        }
    }
    printf("input matrix:\n");
    for(i = 0; i <3; i++){
        for(j = 0; j <3; j++)
            printf("%d\t",N[i][j]);
        printf("\n");
    }
    printf("\n");
    return 0;
}

void main(){
    int x, y, i, j;
    int A[3][3];
    int B[3][3];
    int C[3][3];

    get_3x3(A);
    get_3x3(B);

    for(x = 0; x < 3; x++){
        for(y = 0; y < 3; y++){
            C[x][y] = 0;
            for(i = 0; i < 3; i++){
```

```
            C[x][y] += A[x][i] * B[i][y];
        }
      }
   }
   printf("matrix A*B =\n ");
   for(i = 0; i <3; i++){
      for(j = 0; j < 3; j++)
         printf("%d\t", C[i][j]);
      printf("\n");
   }

}
```

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

답**:**
void (*)(int) signal(int signum, void (* handler)(int))
반환형 **:** **void (*)(int)**
함수 이름**:** **signal**
인자 **:** **int signum, void (* handler)(int)**

9. 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**)**

답**:**
void(*)(void(*handler)(void)) func(int,int)
void(* func(int,int))(void(*handler)(void))
반환형 **:** **void(*)(void(*handler)(void))**
함수 이름**:** **func**
인자 **:** **int, int**

10. 파이프라인이 깨지는 경우 어떠한 이점이 있는지 기술하시오**.**

파이프라인이 깨지면 각 라인에서 미리 실행 준비중이던 명령들이 모두 취소되면서 성능 저하를 불러온다
따라서 이점은 없다

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

```
#include <stdio.h>
double defIntegral(double a, double b){
   double  x, y, dx = 0.000001, sum = 0;
   if(b<a){
      x = b;
      while(x<a){
         y = 4*x*x + 5*x + 1;
         sum += dx*y;
         x += dx;
      }
      return -sum;
   }
   else{
      x = a;

      while(x<b){
         y = 4*x*x + 5*x + 1;
         sum += dx*y;
         x += dx;
      }
      return sum;
   }
}
int main(){
   printf("definite integral : %lf\n",defIntegral(1, 3));
   return 0;
}
```

12. 값이 **1 ~ 4096** 까지 무작위로 할당되어 배열에 저장되도록 프로그래밍 하시오**.**
(배열의 크기는 **100** 개정도로 잡는다**)**

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define ARR_LEN 100
void randArr(unsigned int* arr){
   srand((unsigned int)time(NULL));
   int flag = 0;
   unsigned int ranNum;
   for (int i = 0; i < ARR_LEN; i++){
   retry:
      ranNum = (rand() % 4096) + 1;
      for (int j = i - 1; j >= 0; j--){
         if (arr[i] == arr[j]){
            goto retry;
         }
```

```
        }
        arr[i] = ranNum;
    }
}
int main(){
    unsigned int arr1[ARR_LEN];
    randArr(arr1);
    for (int i = 0; i < ARR_LEN; i++){
        printf("%d ",arr1[i]);
    }
    return 0;
}
```

13. 12 번 문제에서 각 배열은 물건을 담을 수 있는 공간에 해당한다**.**
앞서서 **100** 개의 공간에 물건들을 담았는데 공간의 낭비가 있을 수 있다**.**
이 공간의 낭비가 얼마나 발생했는지 파악하는 프로그램을 작성하시오**.**
?

14. 13 번 문제에서 확장하여 공간을 보다 효율적으로 관리하고 싶어서
4096, 8192, 16384 등의 **4096** 배수로 크기를 확장할 수 있는 시스템을 도입했다**.**
이제부터 공간의 크기는 **4096** 의 배수이고
최소 크기는 **4096,** 최대 크기는 **131072** 에 해당한다**.**
발생할 수 있는 난수는 **1 ~ 131072** 로 설정하고
이를 효율적으로 관리하는 프로그램을 작성하시오**.**
(사실 리눅스 커널의 **Buddy** 메모리 관리 알고리즘임**)**
??

15. 이진 트리를 재귀 호출을 사용하여 구현하도록 한다**.**
(일반적인 **SW** 회사들 면접 당골 문제 - 이게 되면 큐 따위야 문제 없음**)**

22. 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.
(프로그래밍 하시오)

```
#include <stdio.h>
#define ARR_LEN 27
void makeArr(int arr[],int arrLen){
   for(int i = 0; i< arrLen; i++){
      if(i < 2)
         arr[i] = 1;
      else
         arr[i] = arr[i-2] + arr[i-1];
   }
}
void printSum(int arr[],int arrLen){
   int sum1 = 0, sum2 = 0;
   for(int i = 0; i< arrLen; i++){
      if(i%2 == 0)
         sum1 += arr[i];
      else
         sum2 += arr[i];
   }
   printf("홀수들의 합 = %d\n",sum1 );
   printf("짝수들의 합 = %d\n",sum2 );
}
int main(){
   int arr[ARR_LEN];
   makeArr(arr,ARR_LEN);
   printSum(arr,ARR_LEN);
   return 0;
}
```

23. 1, 4, 5, 9, 14, 23, 37, 60, 97, ... 형태로 숫자가 진행된다.
23 번째 숫자는 무엇일까 ?
(프로그래밍 하시오)

```
#include <stdio.h>
#define ARR_LEN 23
void makeArr(int arr[],int arrLen){
   for(int i = 0; i< arrLen; i++){
      if(i == 0)
         arr[i] = 1;
      else if(i == 1)
         arr[i] = 4;
      else
```

```
        arr[i] = arr[i-2] + arr[i-1];
    }
}
int main(){
    int arr[ARR_LEN];
    makeArr(arr,ARR_LEN);
    printf("%d 번째 숫자 = %d\n",ARR_LEN,arr[ARR_LEN-1]);
    return 0;
}
```

24. Intel Architecture 와 **ARM Architecture** 의 차이점은 **?**

25. 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**.**

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void randArr(unsigned int* arr, int totalNum, int loseNum){
    if(loseNum > totalNum){
        printf("불가능!\n");
        return;
        }
    srand((unsigned int)time(NULL));
    int flag = 0;
    unsigned int ranNum;
    for (int i = 0; i < loseNum; i++){
    retry:
        ranNum = (rand() % totalNum) + 1;
        for (int j = i - 1; j >= 0; j--){
            if (arr[i] == arr[j]){
                goto retry;
            }
        }
        arr[i] = ranNum;
    }
    for (int i = 0; i < loseNum; i++){
        printf("%d 번 패배!\n",arr[i]);
```

```
    }
}
int main(){
    unsigned int arr1[10];
    randArr(arr1,25,10);
    return 0;
}
```

26. 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**)**

```
#include <stdio.h>
int square(int x){
    return x*x;
}
void sApply(int arr[2][3], int (*func)(int)){
    for (int i = 0; i < 3; i++){
        arr[1][i] = func(arr[0][i]);
    }
}
int main(){
    int arr[2][3] = {
        {2,4,6},
        {2,4,6}
    };
    sApply(arr,(*square));
    for(int i = 0; i < 2; i++){
        for(int j = 0; j <3 ; j++){
            printf("%d ",arr[i][j]);
        }
        printf("\n");
    }
}
```

27. char *str = "WTF, Where is my Pointer ? Where is it ?" 라는 문자열이 있다
여기에 소문자가 총 몇 개 사용되었는지 세는 프로그램을 만들어보자

```
#include <stdio.h>
#include <string.h>
int smallLetterNum(char* str, int strLen){
   int num = 0;
   for(int i = 0; i < strLen; i++){
      if('a' <= str[i] && str[i] <= 'z')
         num++;
   }
   return num;
}
int main(){
   char str[] = "WTF, Where is my Pointer ? Where is it ?";
   int strLen = strlen(str);
   printf("소문자는 %d 개 \n",smallLetterNum(str,strLen));
   return 0;
}
```

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

답**:**
int *p[3] 이것은 포인터 배열을 의미함
int (*p)[3] 이것은 **int(*)[3]** 크기의 포인터를 의미함

29.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)**

```
#include <stdio.h>
int sort134217728(int x){
   return ~(134217728-1) & x;
}
int main(){
   printf("result = %d\n",sort134217728(999999999));
   return 0;
}
```

30. 단 한 번의 연산으로 대소문자 전환을 할 수 있는 연산에 대해 기술하시오**.**
(프로그래밍 하시오**),** 덧셈 혹은 뺄셈 같은 기능이 아님

```
#include <stdio.h>
#include <string.h>
void changeLetter(char* str, int strLen){
   for(int i = 0; i < strLen; i++){
      str[i] ^= 32;
   }
}
int main(){
   char str[] = "aAbCdeFGhijKLMn";
   int Len = strlen(str);
   changeLetter(str, Len);
   printf("%s\n",str);
   return 0;
}
```

31. 변수의 정의를 기술하시오**.**
답**:** 데이터를 넣을 수 있는 공간

32. 포인터의 정의를 기술하시오**.**
답**:** 변수의 주소값을 넣을 수 있는 공간

33. 함수 포인터의 정의를 기술하시오**.**
답**:** 함수의 주소값을 넣을 수 있는 공간

35. 파이프라인은 언제 깨지는가 **?**
답**:** 다른함수를 불러오거나 점프하는 경우 깨진다

36. 메모리 계층 구조에 대해 기술하시오**.**
속도 빠른 순서**:** 레지스터 - 캐시 - 메모리 - 하드

37. C 언어의 기본 메모리 구조에 대해 기술하시오**.**

38. 우리가 사용하는 윈도우나 리눅스**,** 맥**(**유닉스**)**에서 보는
변수 주소는 진짜 주소일까 아닐까 **?**
알고 있는대로 기술하시오**.**

가짜주소이다

39. 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**)**

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
typedef struct _Data{
   char name[10];
   unsigned int pay;
}Data;
//pay 랜덤으로 주기
void randPay(Data* List, int ListLen){
   srand((unsigned int)time(NULL));
   for(int i = 0; i < ListLen; i++){
      List[i].pay = (rand() % 10) * 10 + 100;
   }
}
void printTopPay(Data* List, int ListLen){
   int index = 0;
   int Top = 0;
   for(int i = 0; i < ListLen; i++){
      if(List[i].pay > Top)
         Top = List[i].pay;
   }
   for(int i = 0; i < ListLen; i++){
      if(List[i].pay == Top){
         printf("%s 의 급여 = %d 만원\n",List[i].name, List[i].pay);
      }
   }
}
int main(){
   //이름 아무거나
   Data List[10] = {
      {"김이름",0},
      {"김급여",0},
      {"김저장",0},
      {"김마음",0},
      {"김고정",0},
      {"김활용",0},
      {"김평균",0},
      {"김가장",0},
      {"김출력",0},
      {"김유의",0}
   };
```

```
    randPay(List,10);
    printTopPay(List, 10);
    return 0;
}
```

44. vi 에서 코드가 정렬이 잘 안되어 있다**.**
이 상황에서 어떻게 해야 예쁘고 깔끔하게 정렬되는가 **?**

답**:** 편집모드에서 드레그**(**마우스 말고 키보드로**)** 후 '=' 누름

51. gdb 를 사용하는 이유를 기술하라**.**
답**:** 컴파일 오류가 아닌 논리적 오류를 찾아내기 위해서

53. a 좌표**(3, 6), b** 좌표**(4, 4)** 가 주어졌다**.**
원점으로부터 이 두 좌표가 만들어내는 삼각형의 크기를 구하는 프로그램을 작성하라**.**

```
#include <stdio.h>
#include <math.h>
typedef struct _Point{
    double x;
    double y;
}Point;

double triArea(Point a, Point b){
    Point o = {0,0};    //원점
    return fabs((o.x * a.y + a.x* b.y + b.x * o.y) - (o.y * a.x + a.y * b.x + b.y * o.x))/2;
}
int main(){
    Point a = {3,2};
    Point b = {0,3};
    printf("삼각형의 넓이 = %lf\n",triArea(a,b));
    return 0;
}
```

54. 가위 바위 보 게임을 만들어보자**.**
프로그램을 만들고 컴퓨터랑 배틀 한다**.**

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
typedef struct _rockScissorPaper{
   char name[10];
}rockScissorPaper;
int main(){
   rockScissorPaper result[3] = {"가위", "바위", "보"};
   srand((unsigned int)time(NULL));
   int tmp;
   int computer = rand() % 3;
   int player;
   printf("가위 -> 0\n 바위 -> 1\n 보  -> 2\n 숫자 입력:");
   scanf("%d",&tmp);
   if(tmp == 0 || tmp == 1 || tmp == 2){
      player = tmp;
   }
   else{
      printf("잘못눌렀습니다\n");
   }

   if( (player + 1) % 3 == computer){
      printf("컴퓨터: %s\n 플레이어: %s\n",result[computer].name,result[player].name);
      printf("결과: 패배!\n");
   }
   else if( (player + 2) % 3 == computer){
      printf("컴퓨터: %s\n 플레이어: %s\n",result[computer].name,result[player].name);
      printf("결과: 승리!\n");
   }
   else{
      printf("컴퓨터: %s\n 플레이어: %s\n",result[computer].name,result[player].name);
      printf("결과: 무승부!\n");
   }
   return 0;
}
```