모든 프로그래밍 문제는 반드시 기능별로 함수를 만들어서 작성하도록 한다.
함수 작성을 통해 기능을 분리하지 않을 경우 70% 감점
(통 함수, 통 메인 전부 감점 대상임)

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 ?

커널

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, 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, 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)

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

편리하고 빠른코드 작성가능 직관적으로 볼수있음
다중루프나 반복문에서 한번에 나오기위해도 사용
현재 수행중인 코드의 위치나 순서에 상관없이 원하는곳으로 강제이동할수있어서

5. 포인터 크기에 대해 알고 있는대로 기술하시오.

32 비트 시스템의 포인터 크기**: 4** 바이트

64 비트 시스템의 포인터 크기: 8 바이트

보편적으로 이렇게 됨 하지만 특수하게 가변해서 사용도 가능

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)

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

```
#include <stdio.h>

void mat_mul(int A[3][3], int B[3][3], int C[3][3]);

int main()

{
int A[3][3] = {{1, 1, 1}, {2, 2, 2}, {3, 3, 3}};
int B[3][3] = {{3, 3, 3}, {2, 2, 2}, {1, 1, 1}};
```

```
int C[3][3] = {0};
int r, c;

mat_mul(A, B, C);
for (r = 0; r < 3; r++)
{
for (c = 0; c < 3; c++)
{
printf("%d ", C[r][c]);
}
printf("\n");
}
return 0;
}

void mat_mul(int A[3][3], int B[3][3], int C[3][3])

{
int r, c, j;
for (r = 0; r < 3; r++)
{
for (c = 0; c < 3; c++)
{
for (j = 0; j < 3; j++)
{
C[r][c] += A[r][j] * B[j][c];
}
}
}
}
```

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

signal 은 인자로 **int signum, void (* handler)(int)**를 인자로 갖고**, (int)**를 인자로갖고
void 를 리턴하는 함수에 대한 포인터를 리턴하는 함수이다
반환형 **void(*)(int)**
함수이름 **signal**
인자 **int signal, 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)

10. 파이프라인이 깨지는 경우 어떠한 이점이 있는지 기술하시오.

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

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

```
#include <stdio.h>

int main(void)
{
int i;
int A[100] = {0}; //정수형 배열 선언
int length = sizeof A / sizeof A[0]; //배열의 길이 지정

for (i = 0; i < length; ++i) //랜덤으로 숫자를 뽑은후 배열에 저장
{
A[i] = rand() % 4096 + 1;
printf("%d ", A[i]);
}
printf("\n");

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)

16. 이진 트리를 재귀 호출 없이 구현하도록 한다.
결과를 확인하는 print 함수(전위, 중위, 후위 순회) 또한 재귀 호출을 수행하면 안됨

17. AVL 트리를 재귀 호출을 사용하여 구현하도록 한다.

18. AVL 트리를 재귀 호출 없이 구현하도록 한다.

19. Red Black 트리와 AVL 트리를 비교해보도록 한다.

20. 난수를 활용하여 Queue 를 구현한다.
(중복되는 숫자를 허용하지 않도록 프로그래밍 하시오)
제일 좋은 방법은 배열을 16 개 놓고 rand() % 16 을 해서
숫자가 겹치지 않는지 확인하면 된다.

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

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)

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

24. Intel Architecture 와 ARM Architecture 의 차이점은 ?

CISC 구조 와 **RISC** 구조
CISC - 모든 가능한 경우를 고려해서 수백개의 명령어를 가지고 있다**.**
RISC - 자주 사용하는 명령어 약 **40** 개 정도만 사용하여 절약된 트랜지스터들로 성능을 향상 시킨다**.**

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.

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)

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>

void Small(char *str);

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

void Small(char *str) // 소문자 개수 출력
{
int cnt = 0;
int i, j;

for (i = 0;; i++)
{
if (str[i] >= 'a' && str[i] <= 'z')
cnt++;
else if (str[i] == '\0')
break;
}
printf("소문자개수: %d \n", cnt);
}
```

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

int *p[3] = 포인터배열 **,** 배열의 요소가 포인터인 배열 **(** 포인터 **3** 개저장
int (*p)[3] = 배열포인터 **,** 포인터가 배열의 주소를 가지게됨 **(**요소 **3** 개가 배열저장된 주소를 저장

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>
```

```c
int main(void)
{
printf("134217728 의 배수=%d\n", ~(134217728-1) & 999999999 );
return 0;
}
```

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

```c
#include <stdio.h>
int main(void)
{
char chr;
scanf("%c", &chr);
printf("chr = %c\n", chr ^ 32); //아스키코드 값참조 10 진수값으로 32 차이남
return 0;
}
```

31. 변수의 정의를 기술하시오.

변하는수 = 자료를 저장할수 있는 방같은 개념

32. 포인터의 정의를 기술하시오.
주소를 저장할 수 있는 변수
무언가를 가르키는 녀석

33. 함수 포인터의 정의를 기술하시오.
데이터 값을 가리키는 대신, 함수 포인터는 메모리 내에서 실행 가능한 코드를 가리킨다
간접호출

34. 재귀호출을 사용하여 queue 를 구현하고 10, 20 을 집어넣는다.

enqueue 과정에 대한 기계어 분석을 수행하여 동작을 파악하도록 한다.
그림과 함께 자세하게 설명하시오.

35. 파이프라인은 언제 깨지는가 ?

36. 메모리 계층 구조에 대해 기술하시오.

레지스터 //위에 있을수록 빠르고,비싸다,용량이적다
캐시 //아래일수록 성능이 줄고 가격이 싸고 용량이 크다
메모리
하드디스크

37. C 언어의 기본 메모리 구조에 대해 기술하시오.

스택 ; 지역변수와 함께 호출시 매개변수 저장
힙 : 동적할당을 할시 힙영역에 저장
데이터 : 전역변수, 정적변수등이 저장
코드: 실행되는 프로그램의 코드부분이 저장

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

가짜주소 , 수업시간에 가짜주소라고 하셨음

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)

40. 리눅스에서 디버깅을 수행하기 위한 프로그램 컴파일 방법을 기술하시오.
우선 컴파일 시에 디버깅 정보를 담아야 한다.
gcc -g -o [프로그램명] [소스파일명]
디버깅 옵션인 -g 으로 컴파일하며,

gdb [프로그램명]
gdb [프로그램명] [core 파일명]
gdb [프로그램명] [실행중인프로세스 pid]

41. 난수를 활용해서 Stack 을 구성한다.
같은 숫자가 들어가지 않게 하고 20 개를 집어넣는다.
이때 들어가는 숫자는 1 ~ 100 사이의 숫자로 넣는다.
(마찬가지로 중복되지 않게 한다)

42. 41 번에서 홀수만 빼내서 AVL 트리를 구성하도록 한다.

43. 41 번에서 짝수만 빼내서 RB 트리를 구성하도록 한다.

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

V 로 블록지정후 =
:gg=G 입력

45. 프로그램을 최적화하는 컴파일 옵션을 적고
반대로 어떠한 최적화도 수행하지 않는 컴파일 옵션을 적도록 한다.

`-O2` 범위내에서 최적화
`-O0` 최적화를 수행하지 않는다

46. 최적화 프로세스를 기술하도록 한다.

소스코드를 기반으로 공통부분과 불필요한 부분을제거
메모리 로드의 기준에서 가장효율적인 메모리나 레지스터의 할당을 하도록함

47. 기존에는 자료구조에서 숫자값만을 받았다.
이제 Queue 에서 데이터로서 숫자가 아닌 문자열을 받아보자.

48. Binary Tree 에서 위 작업을 수행해보라.

49. AVL Tree 에서 위 작업을 수행해보라.

50. 성적 관리 프로그램을 만들어보자.
1. 통계 기능(총 합산, 평균, 표준 편차 계산)
2. 성적순 정렬 기능
3. 성적 입력 기능
4. 학생 정보 삭제 기능

51. gdb 를 사용하는 이유를 기술하라.

스택의 값을 확인하기 편하기 때문

52. 기계어 레벨에서 스택을 조정하는 명령어는 어떤것들이 있는가 ?

Push , pop

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

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

```
/*
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void)
{
int input, a;
int draw = 0, win = 0;

printf("바위는 1, 가위는 2, 보는 3: ");
scanf("%d", &input);
srand((int)time(NULL));                    //srand()와 time()을 이용해서 1~3 까지 난수생성
a = rand() % ((3 - 1) + 1) + 1;

//비겼을 때
if (input == a)
{

//난수가 바위이면
if (a == 1)
{
```

```
printf("나 바위, 컴퓨터바위 , 비겼습니다.");
}

//난수가 가위이면
else if (a == 2)
{
printf("나 가위, 컴퓨터 가위 , 비겼습니다.");
}
//난수가 보이면
else
{
printf("나 보, 컴퓨터 보, 비겼습니다.");
}
}

//사용자가 이겼을 때
else if (input > a)
{

if (input == 1 && a == 2)
{
printf("나 바위, 컴퓨터 가위 , 이겼습니다.");
}

else if (input == 2 && a == 3)
{
printf("나 가위, 컴퓨터 보 , 이겼습니다.");
}

else
{
printf("나 보, 컴퓨터 바위 , 이겼습니다.");
}
}

//사용자가 졌을 때
else
{
if (input == 1 && a == 3)
{
printf(" 바위, 컴퓨터는 보 , 졌습니다.");
}
else if (input == 2 && a == 1)
{
printf("가위, 컴퓨터 바위 , 졌습니다.");
}
else
{
printf("보, 컴퓨터 가위 , 졌습니다.");
}
}
```

```
printf("\n\n");
return 0;
}
```

55. 화면 크기가 가로 800, 세로 600 이다.
여기서 표현되는 값은 x 값이 [-12 ~ + 12] 에 해당하며 y 값은 [-8 ~ +8] 에 해당한다.
x 와 y 가 출력하게 될 값들을 800, 600 화면에 가득차게 표현할 수 있는
스케일링 값을 산출하는 프로그램을 작성하도록 한다.

56. 등차 수열의 합을 구하는 프로그램을 for 문을 도는 방식으로 구현하고
등차 수열의 합 공식을 활용하여 구현해본다.
함수 포인터로 각각의 실행 결과를 출력하고
이 둘의 결과가 같은지 여부를 파악하는 프로그램을 작성하라.

57. sin(x) 값을 프로그램으로 구현해보도록 한다.
어떤 radian 값을 넣던지 그에 적절한 결과를 산출할 수 있도록 프로그래밍 한다.