int main (argc, argv)

- 예외처리
- 동적할당
- 정수변환
- 계산
- 배열 출력
- 할당해제

int ft_handle_exception (char * str, int argc)

예외처리 하는 함수

입력 : 인자로 들어오는 문자열, 인자 개수

return : 예외처리 결과
0 : Error
1 : Success

예외
- 배열길이가 != 16일 때
- 정수가 1 ≤ ≤4 범위인지
- argc != 2
- " " 공백이 아닌 경우

} 출력

void ft_free (udlr, ans)

메모리 할당 해제하는 함수

void ft_atoi (char * str)

문자열 자르고 정수 배열에 저장하는 함수

입력 : 인자로 들어오는 문자열
"col1up col2up ..."

return : 배열의 주소값

void ft_print_box (int ** ans)

답을 출력하는 함수

| v | v | v | /n
| v | v | v | /n
| v | v | v | /n
| v | v | v | /n

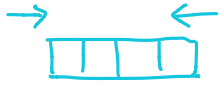
int ft_stack_box (int index , int ** udlr , int * ans)

상자 쌓는 함수 (재귀로 구현)

→ 올바른 자리인지 확인 : ft_check (index , udlr, ans)

void ft_check (int index , int ** udlr , int * ans)

상자 개수가 맞는지 확인하는 함수



- 상자 개수를 udlr과 비교
- 중복이 있는지 확인

int ft_1234check(int index, int *ans)

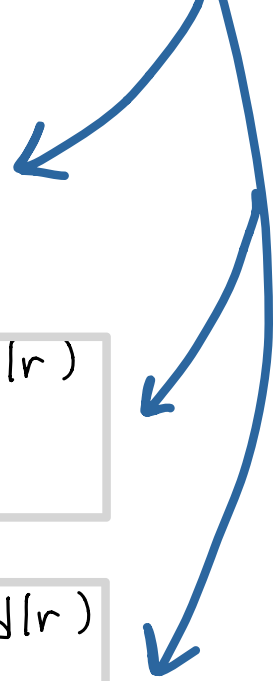
중복 없는지 체크하는 함수
(1,2,3,4)가 돼야함

int row_check(int index, int *ans, int **udlr)

행을 다 채우면 상자 개수 확인하는 함수

int column_check(int index, int *ans, int **udlr)

열을 다 채우면 상자 개수 확인하는 함수



```
int main ( int argc, char* argv[] )
```

메인함수

```
int result
int ** udlr
int * ans
```

입력 예외처리

```
if (ft_handle_exception (argc, argv[1]) == 0)
    return 0
```

↓

```
ans = (int *)malloc ( sizeof(int) * 16 )
```

입력 정수로 변경해서 저장

```
udlr = ft_atoi( argv[1], udlr )
```

↓

계산

```
result = ft_stack_box (0)
```

↓

배열 출력

```
if (result)
    ft_print_box (ans)
else
    write ("Error")
```

↓

할당 해제

```
ft_free ( udlr, ans)
```