brain language

think tanks

```
// #ifdef USE_MOCKS
// #define TEXT "123\0";
// #define fopen(...) (_fopen())
// #define fseek(...) (_fseek())
// #define ftell(...) (_ftell())
// #define fread(...) (_fread(__VA_ARGS__))
// #define fclose(...) (_fclose(__VA_ARGS__))
// #define FILE char
// FILE* _fopen() { return TEXT; }
// int _fseek() { return 0; }
// long _ftell() { return strlen(_fopen()); }
// unsigned long _fread(u8* ptr, size_t size, size_t n, FILE* stream) {
sscanf(stream, "%s", ptr); return _ftell(); }
// void _fclose(FILE* file) { printf("file closed: %s\n", file); }
// #endif

// void process(u8 *data) {
//     printf("%s\n", data);
// }

/**

void process(u8 *data) {
    printf("%s", data);
}

void call(u8* (*_alloc)(), void (*_free)(void*),void(*_function)(u8* data))
{
    u8 *data = _alloc();
    _function(data);
    _free(data);
}


void call_calloc() {
    return calloc(1, size + 1);
}

typedef void (*function)();

*/


// #define MAX_MEMORY 0xffff // 64K bytes
```

```
/* private */

// // global allocated memory
// static void** memory = 0;
// static void** ptr = 0;

// static void init() {
//     ptr = memory = calloc(1, MAX_MEMORY);
// }

// static void destroy() {
//     free(memory);
//     ptr = memory = 0;
// }

// static void list_push(void* data) {
//     *ptr++ = data;
// }

// static void* list_pop() {
//     return *--ptr;
// }

// static void* list_peek() {
//     return *(ptr - 1);
// }

// static void* list_push_peek(void* data) {
//     return *ptr++ = data;
// }

// static void* list_peek_push(void* data) {
//     void* tmp = *ptr;
//     *ptr++ = data;
//     return tmp;
// }

// static void list_copy(void* data) {
//     *(ptr + 1) = *ptr;
//     ptr++;
// }

// static void* list_memory(long index) {
//     return *(memory + index);
// }

// typedef void (*func)();
// typedef void* (*get)();
// typedef void (*set)(void*);
// typedef void* (*proc)(void*);
// typedef void (*set_func)(func f);
// typedef func (*get_func)();

// set copy;
```

```
// set push;
// set push_memory;
// get peek;
// get peek_n;
// get pop;
// proc push_peek;
// proc peek_push;

// void init() {
//     list_init();
//     push = list_push;
//     peek = list_peek;
//     pop = list_pop;
//     push_peek = list_push_peek;
//     peek_push = list_peek_push;
//     copy = list_copy;
// }

// void destroy() {
//     push = 0;
//     peek = 0;
//     pop = 0;
//     list_destroy();
// }

// void push_long(long value) {
//     push((void*)value);
// }

// long peek_long() {
//     return (long)peek();
// }

// long pop_long() {
//     return (long)pop();
// }

// void push_s(char* value) {
//     push(value);
// }

// char* peek_s() {
//     return (char*)peek();
// }

// char* pop_s() {
//     return (char*)pop();
// }

// void push_file(FILE* file) {
//     push(file);
// }

// FILE* peek_file() {
```

```
// 	return (FILE*)peek();
// }

// FILE* pop_file() {
// 	return (FILE*)pop();
// }

// void* pop_long_pop_long_calloc() {
// 	return calloc(pop_long(), pop_long());
// }

// void* pop_s_pop_s_fopen() {
// 	return fopen(pop_s(), pop_s());
// }

// void* peek_ftell() {
// 	return (void*)ftell(peek_file());
// }

// void pop_free() {
// 	free(pop());
// }

// void pop_fclose() {
// 	fclose(pop_file());
// }

// void peek_pop_pop_fseek() {
// 	fseek(peek_file(), pop_long(), (int)pop_long());
// }

//
// int main() {
// 	FILE *f = fopen("input.txt", "rb");
// 	if (f == 0) {
// 		perror("file not found");
// 		exit(1)
// 	} else {
// 		print("all ok")
// 	}
// 	fseek(f, 0, SEEK_END);
// 	u32 size = (u32)ftell(f);
// 	fseek(f, 0, SEEK_SET);

// 	u8 *data = calloc(1, size + 1);
// 	fread(data,1,size,f);
// 	fclose(f);

// 	process(data);
// 	free(data);

// 	return 0;
// }
// */
```

```
// void process() {
//     push(0, "f");
//     push(1, "size");
//     push(2, "data");
//     push(3, SEEK_START);
//     push(4, SEEK_END);
//     push(5, "rb");
//     push(6, "input.txt");
//     push(7, SEEK_SET);
//     push(0, pop_s_pop_s_fopen()); // FILE *f = fopen("input.txt", "rb");
//     peek_pop_pop_fseek();// fseek(f, 0, SEEK_END);
//     push(peek_ftell()); // u32 size = (u32)ftell(f);
//     // fseek(f, 0, SEEK_SET);
// }

// stdio.br

// 2>SEEK_END

// value:
//     <string-value>
//     <integer-value>
//
// expression definitions:
//     <expression>
//
// expr: term ('+' term | '-' term)*
// term: atom ('*' atom | '/' atom)*
//
// variable defintions:
//     <expression>'> '<variable>
//
// function definitions (not internal):
//     <function>': '<expressions>
//
// functions calls:
//     <function>'< '<variables>
//
// if statements:
//     '? '<condition>': '<statements>['! '<statements>]' '
//
// while statement
//     '? '<condition>': '<statements>' <'
//
// do-while statement
//     ': '<statements>'? '<condition>' <'
//
// condition definitions:
//     <expression>'=='<expression>
//     <expression>'!='<expression>
//     <expression>'>='<expression>
//     <expression>'<='<expression>
//     <expression>'>'<expression>
```

```
//     <expression>'<'<expression>
//
//
//statement: assignment | expr | if_statement
//expr: expr '+' term | expr '-' term | term
//term: term '*' atom | term '/' atom | atom
//atom: NAME | NUMBER | '(' expr ')'
//assignment: target '=' expr
//target: NAME
//if_statement: 'if' expr ':' statement
//
//
//
//
//
//
// fopen<"input.txt","rb">f
//     ? f==0
//         perror<"file not found"
//         exit<1
//     :
//         print<"all ok"
//     !
//
// translates to
//
// fopen<"input.txt","rb">f (? f==0 perror<"file not found" exit<1 :
print<"all ok")
//
// == .. ?, < .. ?
//
// () are the same as grouping, cause no () in function calls
//
// fopen<"input.txt","rb">f ? f==0: perror<"file not found" exit<1!:
print<"all ok"

// fopen<"input.txt","rb">f ? f==0: perror<"file not found" exit<1 :
fseek<f,0,SEEK_END ftell(f)>size fseek(f, 0, SEEK_SET) // name: means
function definition with name 'name'
// f>fclose
// process<data

// int main() {
//     init();
//     // process();
//     destroy();
// }
```