iPad:~/Study/Labs/sem2labs/26# cat stack.h

```
#define INIT_SIZE 10

#ifndef _STACK_H_
#define _STACK_H_

#include <stdlib.h>

#define STACK_OVERFLOW  -100
#define STACK_UNDERFLOW -101
#define OUT_OF_MEMORY   -102
#define MULTIPLIER 2

typedef int data_type;

typedef struct {
    data_type *data;
    size_t size;
    size_t top;
} Stack;

Stack* stack_create(void)
{
    Stack *out = NULL;
    out = malloc(sizeof(Stack));
    if (out == NULL) {
        exit(OUT_OF_MEMORY);
    }
    out->size = INIT_SIZE;
    out->data = malloc(out->size * sizeof(data_type));
    if (out->data == NULL) {
        free(out);
        exit(OUT_OF_MEMORY);
    }
    out->top = 0;
    return out;
}

void stack_delete (Stack **stack) {
    free((*stack)->data);
    free(*stack);
    *stack = NULL;
}

void resize(Stack *stack) {
    stack->size *= MULTIPLIER;
    stack->data = realloc(stack->data, stack->size * sizeof(data_type));
    if (stack->data == NULL) {
        exit(STACK_OVERFLOW);
    }
}

int stack_is_empty(Stack *stack)
{
return stack->top == 0;
}

void stack_push(Stack *stack, data_type value)
{
        if (stack->top >= stack->size) {
```

```
        resize(stack);
    }
    stack->data[stack->top] = value;
    stack->top++;
}

data_type stack_pop(Stack *stack)
{
        if (stack->top == 0) {
        exit(STACK_UNDERFLOW);
    }
    stack->top--;
    return stack->data[stack->top];
}

data_type peek(Stack *stack) {
    if (stack->top <= 0) {
        exit(STACK_UNDERFLOW);
    }
    return stack->data[stack->top - 1];
}

void stack_print(Stack *stack)
{
        for(int i = 0; i + 1 <= stack->top; i++)
        {
                printf("%d\n", stack->data[i]);
        }
}

size_t stack_size(Stack *stack)
{
        return stack->top;
}

void stack_concatenation(Stack *A, Stack *B)
{
        Stack *T = stack_create();
        while(!stack_is_empty(B)) {
                stack_push(T, stack_pop(B));
        }
        while(!stack_is_empty(T)) {
                stack_push(A, stack_pop(T));
        }
        stack_delete(&T);
}

#endif
iPad:~/Study/Labs/sem2labs/26# cat hoar_sort.c
#include<stdio.h>

#include"stack.h"

void sort(Stack *A);

int main(void)
{
        int a;
        Stack *A = stack_create();
        while(scanf("%d", &a) == 1) {
```

```
            stack_push(A, a);
        }
        putchar('\n');
        puts("-------------------------");
        stack_print(A);
        puts("-------------------------");
        sort(A);
        stack_print(A);
        return 0;
}


void sort(Stack *A)
{
        int a;
        int key = stack_pop(A);
        Stack *L = stack_create();
        Stack *G = stack_create();
        while(!stack_is_empty(A)) {
                a = stack_pop(A);
                if ( a < key) {
                        stack_push(L, a);
                } else {
                        stack_push(G, a);
                }
        }
        if (stack_size(L) > 1) {
                sort(L);
        }
        if (stack_size(G) > 1) {
                sort(G);
        }
        stack_push(L, key);
        stack_concatenation(L, G);
        stack_concatenation(A, L);
        stack_delete(&L);
        stack_delete(&G);
}
iPad:~/Study/Labs/sem2labs/26# gcc -Wall -pedantic -
std=c99 hoar_sort.c
iPad:~/Study/Labs/sem2labs/26# ./a.out
9 4 12 3 5 66 2 3 44 24 -5 -12 0 3 1 37

-------------------------
9
4
12
3
5
66
2
3
44
24
-5
-12
0
3
1
37
```

```
------------------------
-12
-5
0
1
2
3
3
3
4
5
9
12
24
37
44
66
iPad:~/Study/Labs/sem2labs/26#
```