```c
/**
* Lab 8 - Q. 2 - GVSU
*
* Experimenting with memory management.
*
* @author Ron Rounsifer
**/
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>

int init_global = 5235;
int uninit_global;
int g_array_uninit[50];
int g_array_init[3] = {1,2,3};


// add a struct
struct node {
    int value;
};


int main() {

    // playground
    // Experiment with different types of variable (global, local, initialized, unit
ialized, arrays, pointers, structures, etc...)

    char *l_ptr = malloc(sizeof(char));
    int local = 555;
    int uninit_local;
    int fact = factorial(5);
    struct node test;

    printf("Code\n");
    printf("factorial is stored at: %p\n\n", factorial);


    printf("Stack\n");
    printf("local init stored at:    %p\n", &local);
    printf("local uninit stored at:    %p\n", &uninit_local);
    printf("local pointer stored at:   %px\n", &l_ptr);
    printf("fact stored at: %p\n", &fact);
    function_variable();
    printf("node structure located at: %p\n\n", &test);

    printf("\n\n");
    printf("Heap\n");
    printf("Run time data:\n");
    printf("local pointers data stored at:   %p\n\n", l_ptr);

    printf("Run time libraries:\n");
    printf("printf() function located at: %p", printf);

    printf("\n\n");
    printf("Data\n");
    printf("global array uninit:   %p\n",  &g_array_uninit);
    printf("uninit global:         %p\n",  &uninit_global);

    printf("\n\n");
    printf("Init Data\n") ;
    printf("init global:    %p\n",  &init_global);
    printf("global array init:   %p\n", &g_array_init);

    printf("\n");



return 0;
}
```

```c
void function_variable()
{
    int function = 2483;
    printf("function stack:      %p\n",   &function);
};

int factorial(int n) {
    if (n == 0)
    {
        return n;
    }
    return n * factorial(n-1);
}
```