

Exercises for 01-dynamic-memory

Dynamic-Memory

In some of the following exercises we refer to a struct called person which you find here below:

```
typedef struct person_ {  
    // Hmmm, not good enough to have fixed name sizes  
    char first_name[20];  
    char family_name[20];  
} person ;
```

In the suggested solutions below we will use the String function without n (e.g. strcpy) instead of the n-ones (e.g. strncpy) when we know that nothing can go wrong.

0.

Write a program that allocates 1000 bytes, using malloc and stores a reference to that memory in a variable.

Note: your program now leaks memory

1.

Write a program that allocates 100000000000 bytes, using malloc and stores a reference to that memory in a variable.

Note: You will most likely not be able to allocate this amount of memory. Make sure to detect a mem allocation failure.

2.

Fix the memory leak in (1).

Hint: Use the function free

3.

Write a program that allocates memory for 1 person, using calloc.

Note: your program now leaks memory

4.

Fix the memory leak in (3).

Hint: Use the function free

5.

Write a program that allocates memory for 1 person, using calloc. Add a name to the person. Print the name out.

After the print statement in you should add code the uses the function realloc to increase (in size) the memory, previously allocated, by 1. The memory should now be of a size that can store two persons.

Give the second person a name as well.

Print both persons out.

6.

Allocate memory for 100 persons. Add names (can be the same name) to each person using a loop.

Now you should assume there are 100 more persons and give them names as well. This will lead to you using accessing memory you have not allocated.

Compile and run the program.

Note: If you're using GNU/Linux (e.g. Debian, Ubuntu, RedHat, Fedora...) or MacOS try out the program valgrind. If you're running Windows you can try valgrind on Ubuntu using VirtualBox.

7.

Write a program that allocates memory for one more person structs in a loop and gives them names.

Pseudo code:

```
for (i=0;i<100;i++) {  
    increase memory with size for one person (use realloc)  
    give the person at pos i a name  
}
```

Make sure to free the allocated memory.

8.

Redo the exercises above with the following person definition instead.

```
typedef struct person_ {  
    char *first_name;  
    char *family_name;  
} person ;
```