0x0C. C - More malloc, free

CMemory allocation

- By: Julien Barbier
- Weight: 1
- Project over - took place from Apr 7, 2022 6:00 AM to Apr 8, 2022 6:00 AM
- An auto review will be launched at the deadline

In a nutshell…

- **Auto QA review:** 36.0/36 mandatory & 15.0/21 optional
- **Altogether: 171.43%**
  - Mandatory: 100.0%
  - Optional: 71.43%
  - Calculation: 100.0% + (100.0% * 71.43%) == **171.43%**

Concepts

For this project, we expect you to look at this concept:

- Automatic and dynamic allocation, malloc and free

Resources

Read or watch:

- Do I cast the result of malloc?

man or help:

- `exit (3)`
- `calloc`
- `realloc`

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, **without the help of Google**:

General

- How to use the `exit` function

- What are the functions `calloc` and `realloc` from the standard library and how to use them

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else's work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be compiled on Ubuntu 20.04 LTS using `gcc`, using the options `-Wall -Werror -Wextra -pedantic -std=gnu89`
- All your files should end with a new line
- A `README.md` file, at the root of the folder of the project is mandatory
- Your code should use the `Betty` style. It will be checked using betty-style.pl and betty-doc.pl
- You are not allowed to use global variables
- No more than 5 functions per file
- The only C standard library functions allowed are `malloc`, `free` and `exit`. Any use of functions like `printf`, `puts`, `calloc`, `realloc` etc… is forbidden
- You are allowed to use `_putchar`
- You don't have to push `_putchar.c`, we will use our file. If you do it won't be taken into account
- In the following examples, the `main.c` files are shown as examples. You can use them to test your functions, but you don't have to push them to your repo (if you do we won't take them into account). We will use our own `main.c` files at compilation. Our `main.c` files might be different from the one shown in the examples
- The prototypes of all your functions and the prototype of the function `_putchar` should be included in your header file called `main.h`
- Don't forget to push your header file

Quiz questions

Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. Trust no one

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that allocates memory using `malloc`.

- Prototype: `void *malloc_checked(unsigned int b);`
- Returns a pointer to the allocated memory
- if `malloc` fails, the `malloc_checked` function should cause normal process termination with a status value of `98`

```
julien@ubuntu:~/0x0b. more malloc, free$ cat 0-main.c
#include "main.h"
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char *c;
    int *i;
    float *f;
    double *d;

    c = malloc_checked(sizeof(char) * 1024);
    printf("%p\n", (void *)c);
    i = malloc_checked(sizeof(int) * 402);
    printf("%p\n", (void *)i);
    f = malloc_checked(sizeof(float) * 100000000);
    printf("%p\n", (void *)f);
    d = malloc_checked(INT_MAX);
    printf("%p\n", (void *)d);
    free(c);
```

```
    free(i);
    free(f);
    free(d);
    return (0);
}
julien@ubuntu:~/0x0b. more malloc, free$ gcc -Wall -pedantic -Werror -Wextra -std=gnu
89 0-main.c 0-malloc_checked.c -o a
julien@ubuntu:~/0x0b. more malloc, free$ ./a
0x1e39010
0x1e39830
0x7f31f6c19010
julien@ubuntu:~/0x0b. more malloc, free$ echo $?
98
julien@ubuntu:~/0x0b. more malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0C-more_malloc_free`
- File: `0-malloc_checked.c`

Done! Help Check your code QA Review

1. string_nconcat

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that concatenates two strings.

- Prototype: `char *string_nconcat(char *s1, char *s2, unsigned int n);`
- The returned pointer shall point to a newly allocated space in memory, which contains `s1`, followed by the first `n` bytes of `s2`, and null terminated
- If the function fails, it should return `NULL`
- If `n` is greater or equal to the length of `s2` then use the entire string `s2`
- if `NULL` is passed, treat it as an empty string

```
julien@ubuntu:~/0x0b. more malloc, free$ cat 1-main.c
#include "main.h"
#include <stdio.h>
#include <stdlib.h>
```

```
/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char *concat;

    concat = string_nconcat("Best ", "School !!!", 6);
    printf("%s\n", concat);
    free(concat);
    return (0);
}
julien@ubuntu:~/0x0b. more malloc, free$ gcc -Wall -pedantic -Werror -Wextra -std=gnu
89 1-main.c 1-string_nconcat.c -o 1-string_nconcat
julien@ubuntu:~/0x0b. more malloc, free$ ./1-string_nconcat
Best School
julien@ubuntu:~/0x0b. more malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0C-more_malloc_free`
- File: `1-string_nconcat.c`

Done! Help Check your code QA Review

2. _calloc

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that allocates memory for an array, using `malloc`.

- Prototype: `void *_calloc(unsigned int nmemb, unsigned int size);`
- The `_calloc` function allocates memory for an array of `nmemb` elements of `size` bytes each and returns a pointer to the allocated memory.
- The memory is set to zero
- If `nmemb` or `size` is `0`, then `_calloc` returns `NULL`

- If `malloc` fails, then `_calloc` returns `NULL`

FYI: The standard library provides a different function: `calloc`. Run `man calloc` to learn more.

```
julien@ubuntu:~/0x0b. more malloc, free$ cat 2-main.c
#include "main.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/**
 * simple_print_buffer - prints buffer in hexa
 * @buffer: the address of memory to print
 * @size: the size of the memory to print
 *
 * Return: Nothing.
 */
void simple_print_buffer(char *buffer, unsigned int size)
{
    unsigned int i;

    i = 0;
    while (i < size)
    {
        if (i % 10)
        {
            printf(" ");
        }
        if (!(i % 10) && i)
        {
            printf("\n");
        }
        printf("0x%02x", buffer[i]);
        i++;
    }
```

```
    printf("\n");
}


/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    char *a;


    a = _calloc(98, sizeof(char));
    strcpy(a, "Best");
    strcpy(a + 4, " School! :)\n");
    a[97] = '!';
    simple_print_buffer(a, 98);
    free(a);
    return (0);
}
julien@ubuntu:~/0x0b. more malloc, free$ gcc -Wall -pedantic -Werror -Wextra -std=gnu
89 2-main.c 2-calloc.c -o 2-calloc
julien@ubuntu:~/0x0b. more malloc, free$ ./2-calloc
0x42 0x65 0x73 0x74 0x20 0x53 0x63 0x68 0x6f 0x6f
0x6c 0x21 0x20 0x3a 0x29 0x0a 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x21
```

```
julien@ubuntu:~/0x0b. more malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0C-more_malloc_free`
- File: `2-calloc.c`

Done! Help Check your code QA Review

3. array_range

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that creates an array of integers.

- Prototype: `int *array_range(int min, int max);`
- The array created should contain all the values from `min` (included) to `max` (included), ordered from `min` to `max`
- Return: the pointer to the newly created array
- If `min` > `max`, return `NULL`
- If `malloc` fails, return `NULL`

```
julien@ubuntu:~/0x0b. more malloc, free$ cat 3-main.c
#include "main.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/**
 * simple_print_buffer - prints buffer in hexa
 * @buffer: the address of memory to print
 * @size: the size of the memory to print
 *
 * Return: Nothing.
 */
void simple_print_buffer(int *buffer, unsigned int size)
{
    unsigned int i;
```

```
    i = 0;
    while (i < size)
    {
        if (i % 10)
        {
            printf(" ");
        }
        if (!(i % 10) && i)
        {
            printf("\n");
        }
        printf("0x%02x", buffer[i]);
        i++;
    }
    printf("\n");
}


/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    int *a;


    a = array_range(0, 10);
    simple_print_buffer(a, 11);
    free(a);
    return (0);
}
julien@ubuntu:~/0x0b. more malloc, free$ gcc -Wall -pedantic -Werror -Wextra -std=gnu
89 3-main.c 3-array_range.c -o 3-array_range
julien@ubuntu:~/0x0b. more malloc, free$ ./3-array_range
```

```
0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09
0x0a
julien@ubuntu:~/0x0b. more malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0C-more_malloc_free`
- File: `3-array_range.c`

Done! Help Check your code QA Review

4. _realloc

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that reallocates a memory block using `malloc` and `free`

- Prototype: `void *_realloc(void *ptr, unsigned int old_size, unsigned int new_size);`
- where `ptr` is a pointer to the memory previously allocated with a call to `malloc`: `malloc(old_size)`
- `old_size` is the size, in bytes, of the allocated space for `ptr`
- and `new_size` is the new size, in bytes of the new memory block
- The contents will be copied to the newly allocated space, in the range from the start of `ptr` up to the minimum of the old and new sizes
- If `new_size` > `old_size`, the “added” memory should not be initialized
- If `new_size` == `old_size` do not do anything and return `ptr`
- If `ptr` is `NULL`, then the call is equivalent to `malloc(new_size)`, for all values of `old_size` and `new_size`
- If `new_size` is equal to zero, and `ptr` is not `NULL`, then the call is equivalent to `free(ptr)`. Return `NULL`
- Don’t forget to free `ptr` when it makes sense

FYI: The standard library provides a different function: `realloc`. Run `man realloc` to learn more.

```
julien@ubuntu:~/0x0b. more malloc, free$ cat 100-main.c
#include "main.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/**
 * simple_print_buffer - prints buffer in hexa
```

```
 * @buffer: the address of memory to print
 * @size: the size of the memory to print
 *
 * Return: Nothing.
 */
void simple_print_buffer(char *buffer, unsigned int size)
{
    unsigned int i;

    i = 0;
    while (i < size)
    {
        if (i % 10)
        {
            printf(" ");
        }
        if (!(i % 10) && i)
        {
            printf("\n");
        }
        printf("0x%02x", buffer[i]);
        i++;
    }
    printf("\n");
}

/**
 * main - check the code for
 *
 * Return: Always 0.
 */
int main(void)
{
```

```
    char *p;
    int i;

    p = malloc(sizeof(char) * 10);
    p = _realloc(p, sizeof(char) * 10, sizeof(char) * 98);
    i = 0;
    while (i < 98)
    {
        p[i++] = 98;
    }
    simple_print_buffer(p, 98);
    free(p);
    return (0);
}
julien@ubuntu:~/0x0b. more malloc, free$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 100-main.c 100-realloc.c -o 100-realloc
julien@ubuntu:~/0x0b. more malloc, free$ ./100-realloc
0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62
0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62
0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62
0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62
0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62
0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62
0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62
0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62
0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62
0x62 0x62 0x62 0x62 0x62 0x62 0x62 0x62
julien@ubuntu:~/0x0b. more malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0C-more_malloc_free`
- File: `100-realloc.c`

Done! Help Check your code QA Review

5. We must accept finite disappointment, but never lose infinite hope

#advanced

Score: 40.0% (*Checks completed: 40.0%*)

Write a program that multiplies two positive numbers.

- Usage: `mul num1 num2`
- `num1` and `num2` will be passed in base 10
- Print the result, followed by a new line
- If the number of arguments is incorrect, print `Error`, followed by a new line, and exit with a status of `98`
- `num1` and `num2` should only be composed of digits. If not, print `Error`, followed by a new line, and exit with a status of `98`
- You are allowed to use more than 5 functions in your file

You can use `bc` (`man bc`) to check your results.

```
julien@ubuntu:~/0x0b. more malloc, free$ gcc -Wall -pedantic -Werror -Wextra -std=gnu
89 101-mul.c _putchar.c -o 101-mul

julien@ubuntu:~/0x0b. more malloc, free$ ./101-mul 10 98

980

julien@ubuntu:~/0x0b. more malloc, free$ ./101-mul 2352346932694364362234465265463345
764376347653786538758746876496986595866958985 79 2865803436508436508342608310967913760
821640863143081430865108465081640613406083160831085308610376901370967506713058657083 2
760732096730978014607369739567864508634086304807450973045703428580934825098342095832 4
098503942850983425098342095834253452674136923575589187997046452422615907476091498993
541335055687577080701989306920124712185512283638941702255216631601001307425878158314 3
870461182707893577849408672040555089482160343085482612348145322689883025225988799452 3
292902811699275321605906519935117885185505475702845747159250069627382628886178404353 8
9140329668772644708

674136392357558918799704645242261590747609149899354133505568757708070198930692012471 2
185512283638941702255216631601001307425878158314387046118270789357784940867204055508 9
482160343085482612348145322689883025225988799452329290281169927532160590810573779266 5
133761261824833211325690248597437196938515601506881386827400068391218781860166705860 5
418678284322237297213673482412392922068159291496274311170208689056585352782844484721 1
408463677416499626386492295092818678960672084741784021562949789407129595183518464138 5
914179238085331381201529533354671663434428408642677548077574780815003073211970486780 5
688704303461042373101473485092019906795014369069932

julien@ubuntu:~/0x0b. more malloc, free$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x0C-more_malloc_free`
- File: `101-mul.c`

Done? Help Check your code Ask for a new correction Get a sandbox QA Review