0x08. C - Recursion

CRecursion

- By: Julien Barbier
- Weight: 1
- Project over - took place from Mar 30, 2022 6:00 AM to Mar 31, 2022 6:00 AM
- An auto review will be launched at the deadline

In a nutshell…

- **Auto QA review:** 49.0/49 mandatory & 24.0/24 optional
- **Altogether: 200.0%**
  - Mandatory: 100.0%
  - Optional: 100.0%
  - Calculation: 100.0% + (100.0% * 100.0%) == **200.0%**

Resources

Read or watch:

- 0x08. Recursion, introduction
- What on Earth is Recursion?
- C - Recursion

- C Programming Tutorial 85, Recursion pt.1
- C Programming Tutorial 86, Recursion pt.2

Learning Objectives

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

General

- What is recursion
- How to implement recursion
- In what situations you should implement recursion
- In what situations you shouldn't implement recursion

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
- You are not allowed to use the standard library. Any use of functions like `printf`, `puts`, 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
- **You are not allowed to use any kind of loops**
- You are not allowed to use `static` variables

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

Tasks

0. She locked away a secret, deep inside herself, something she once knew to be true... but chose to forget

mandatory

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

Write a function that prints a string, followed by a new line.

- Prototype: `void _puts_recursion(char *s);`

FYI: The standard library provides a similar function: `puts`. Run `man puts` to learn more.

```
julien@ubuntu:~/0x08. Recursion$ cat 0-main.c
#include "main.h"


/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    _puts_recursion("Puts with recursion");
    return (0);
}
julien@ubuntu:~/0x08. Recursion$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putc
har.c 0-main.c 0-puts_recursion.c -o 0-puts_recursion
julien@ubuntu:~/0x08. Recursion$ ./0-puts_recursion
```

```
Puts with recursion
julien@ubuntu:~/0x08. Recursion$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x08-recursion`
- File: `0-puts_recursion.c`

Done! Help Check your code Get a sandbox QA Review

1. Why is it so important to dream? Because, in my dreams we are together

mandatory

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

Write a function that prints a string in reverse.

- Prototype: `void _print_rev_recursion(char *s);`

```
julien@ubuntu:~/0x08. Recursion$ cat 1-main.c
#include "main.h"


/**
 * main - check the code
 *
 * Return: Always 0.
 */
int main(void)
{
    _print_rev_recursion("\nColton Walker");
    return (0);
}
julien@ubuntu:~/0x08. Recursion$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 _putc
har.c 1-main.c 1-print_rev_recursion.c -o 1-print_rev_recursion
julien@ubuntu:~/0x08. Recursion$ ./1-print_rev_recursion
reklaW notloC
julien@ubuntu:~/0x08. Recursion$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x08-recursion`
- File: `1-print_rev_recursion.c`

Done? Help Check your code Get a sandbox QA Review

2. Dreams feel real while we're in them. It's only when we wake up that we realize something was actually strange

mandatory

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

Write a function that returns the length of a string.

- Prototype: `int _strlen_recursion(char *s);`

FYI: The standard library provides a similar function: `strlen`. Run `man strlen` to learn more.

```
julien@ubuntu:~/0x08. Recursion$ cat 2-main.c
#include "main.h"
#include <stdio.h>

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

    n = _strlen_recursion("Corbin Coleman");
    printf("%d\n", n);
    return (0);
}
julien@ubuntu:~/0x08. Recursion$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89  2-ma
in.c 2-strlen_recursion.c -o 2-strlen_recursion
julien@ubuntu:~/0x08. Recursion$ ./2-strlen_recursion
14
julien@ubuntu:~/0x08. Recursion$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x08-recursion`
- File: `2-strlen_recursion.c`

Done! Help Check your code Get a sandbox QA Review

3. You mustn't be afraid to dream a little bigger, darling

mandatory

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

Write a function that returns the factorial of a given number.

- Prototype: `int factorial(int n);`
- If `n` is lower than `0`, the function should return `-1` to indicate an error
- Factorial of `0` is `1`

```
julien@ubuntu:~/0x08. Recursion$ cat 3-main.c
#include "main.h"
#include <stdio.h>

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


    r = factorial(1);
    printf("%d\n", r);
    r = factorial(5);
    printf("%d\n", r);
    r = factorial(10);
    printf("%d\n", r);
    r = factorial(-1024);
    printf("%d\n", r);
```

```
    return (0);
}
julien@ubuntu:~/0x08. Recursion$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 3-mai
n.c 3-factorial.c -o 3-factorial
julien@ubuntu:~/0x08. Recursion$ ./3-factorial
1
120
3628800
-1
julien@ubuntu:~/0x08. Recursion$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x08-recursion`
- File: `3-factorial.c`

Done! Help Check your code Get a sandbox QA Review

4. Once an idea has taken hold of the brain it's almost impossible to eradicate

mandatory

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

Write a function that returns the value of `x` raised to the power of `y`.

- Prototype: `int _pow_recursion(int x, int y);`
- If `y` is lower than `0`, the function should return `-1`

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

```
julien@ubuntu:~/0x08. Recursion$ cat 4-main.c
#include "main.h"
#include <stdio.h>

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

```
{
    int r;


    r = _pow_recursion(1, 10);
    printf("%d\n", r);
    r = _pow_recursion(1024, 0);
    printf("%d\n", r);
    r = _pow_recursion(2, 16);
    printf("%d\n", r);
    r = _pow_recursion(5, 2);
    printf("%d\n", r);
    r = _pow_recursion(5, -2);
    printf("%d\n", r);
    r = _pow_recursion(-5, 3);
    printf("%d\n", r);
    return (0);
}
julien@ubuntu:~/0x08. Recursion$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 4-mai
n.c 4-pow_recursion.c -o 4-pow
julien@ubuntu:~/0x08. Recursion$ ./4-pow
1
1
65536
25
-1
-125
julien@ubuntu:~/0x08. Recursion$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x08-recursion`
- File: `4-pow_recursion.c`

Done! Help Check your code Get a sandbox QA Review

5. Your subconscious is looking for the dreamer

mandatory

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

Write a function that returns the natural square root of a number.

- Prototype: `int _sqrt_recursion(int n);`
- If `n` does not have a natural square root, the function should return `-1`

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

```
julien@ubuntu:~/0x08. Recursion$ cat 5-main.c
#include "main.h"
#include <stdio.h>

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

    r = _sqrt_recursion(1);
    printf("%d\n", r);
    r = _sqrt_recursion(1024);
    printf("%d\n", r);
    r = _sqrt_recursion(16);
    printf("%d\n", r);
    r = _sqrt_recursion(17);
    printf("%d\n", r);
    r = _sqrt_recursion(25);
    printf("%d\n", r);
    r = _sqrt_recursion(-1);
    printf("%d\n", r);
    return (0);
}
```

```
julien@ubuntu:~/0x08. gcc -Wall -pedantic -Werror -Wextra -std=gnu89 5-main.c 5-sqrt_
recursion.c -o 5-sqrt
julien@ubuntu:~/0x08. Recursion$ ./5-sqrt
1
32
4
-1
5
-1
julien@ubuntu:~/0x08. Recursion$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x08-recursion`
- File: `5-sqrt_recursion.c`

Done! Help Check your code Get a sandbox QA Review

6. Inception. Is it possible?

mandatory

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

Write a function that returns `1` if the input integer is a prime number, otherwise return `0`.

- Prototype: `int is_prime_number(int n);`

```
julien@ubuntu:~/0x08. Recursion$ cat 6-main.c
#include "main.h"
#include <stdio.h>

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

```
    r = is_prime_number(1);
    printf("%d\n", r);
    r = is_prime_number(1024);
    printf("%d\n", r);
    r = is_prime_number(16);
    printf("%d\n", r);
    r = is_prime_number(17);
    printf("%d\n", r);
    r = is_prime_number(25);
    printf("%d\n", r);
    r = is_prime_number(-1);
    printf("%d\n", r);
    r = is_prime_number(113);
    printf("%d\n", r);
    r = is_prime_number(7919);
    printf("%d\n", r);
    return (0);
}
julien@ubuntu:~/0x08. Recursion$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 6-mai
n.c 6-is_prime_number.c -o 6-prime
julien@ubuntu:~/0x08. Recursion$ ./6-prime
0
0
0
1
0
0
1
1
julien@ubuntu:~/0x08. Recursion$
```

Repo:

- GitHub repository: `alx-low_level_programming`

- Directory: `0x08-recursion`
- File: `6-is_prime_number.c`

Done! Help Check your code Get a sandbox QA Review

7. They say we only use a fraction of our brain's true potential. Now that's when we're awake. When we're asleep, we can do almost anything

#advanced

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

Write a function that returns `1` if a string is a palindrome and `0` if not.

- Prototype: `int is_palindrome(char *s);`
- An empty string is a palindrome

```
julien@ubuntu:~/0x08. Recursion$ cat 100-main.c
#include "main.h"
#include <stdio.h>

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

    r = is_palindrome("level");
    printf("%d\n", r);
    r = is_palindrome("redder");
    printf("%d\n", r);
    r = is_palindrome("test");
    printf("%d\n", r);
    r = is_palindrome("step on no pets");
    printf("%d\n", r);
    return (0);
}
```

```
julien@ubuntu:~/0x08. Recursion$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 100-main.c 100-is_palindrome.c -o 100-palindrome
julien@ubuntu:~/0x08. Recursion$ ./100-palindrome
1
1
0
1
julien@ubuntu:~/0x08. Recursion$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x08-recursion`
- File: `100-is_palindrome.c`

Done! Help Check your code Get a sandbox QA Review

8. Inception. Now, before you bother telling me it's impossible...

#advanced

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

Write a function that compares two strings and returns `1` if the strings can be considered identical, otherwise return `0`.

- Prototype: `int wildcmp(char *s1, char *s2);`
- `s2` can contain the special character `*`.
- The special char `*` can replace any string (including an empty string)

```
julien@ubuntu:~/0x08. Recursion$ cat 101-main.c
#include "main.h"
#include <stdio.h>


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

```
    int r;


    r = wildcmp("main.c", "*.c");
    printf("%d\n", r);
    r = wildcmp("main.c", "m*a*i*n*.*c*");
    printf("%d\n", r);
    r = wildcmp("main.c", "main.c");
    printf("%d\n", r);
    r = wildcmp("main.c", "m*c");
    printf("%d\n", r);
    r = wildcmp("main.c", "ma******************************c");
    printf("%d\n", r);
    r = wildcmp("main.c", "*");
    printf("%d\n", r);
    r = wildcmp("main.c", "***");
    printf("%d\n", r);
    r = wildcmp("main.c", "m.*c");
    printf("%d\n", r);
    r = wildcmp("main.c", "**.*c");
    printf("%d\n", r);
    r = wildcmp("main-main.c", "ma*in.c");
    printf("%d\n", r);
    r = wildcmp("main", "main*d");
    printf("%d\n", r);
    r = wildcmp("abc", "*b");
    printf("%d\n", r);
    return (0);
}
julien@ubuntu:~/0x08. Recursion$ gcc -Wall -pedantic -Werror -Wextra -std=gnu89 101-m
ain.c 101-wildcmp.c -o 101-wildcmp
julien@ubuntu:~/0x08. Recursion$ ./101-wildcmp
1
1
1
```

```
1
1
1
1
0
1
1
0
0
julien@ubuntu:~/0x08. Recursion$
```

Repo:

- GitHub repository: `alx-low_level_programming`
- Directory: `0x08-recursion`
- File: `101-wildcmp.c`

Done! Help Check your code Get a sandbox QA Review