You just released the advanced tasks of this project. Have fun!

0x11. C - printf

Concepts

For this project, we expect you to look at these concepts:

- Group Projects (/concepts/111)
- Pair Programming How To (/concepts/121)
- Flowcharts (/concepts/130)
- Technical Writing (/concepts/225)

Background Context

Write your own printf function.



^ In this picture, Kris (/rltoken/pSPZEmqi5O8ZoeLM5-65WA), and Jul (/rltoken/X_vDffLlUpbtqnubfnQx8Q)

Resources

Read or watch:

- Secrets of printf (/rltoken/7Vw7aUWgwC7JYUrql4bh4Q)
- Group Projects concept page (Don't forget to read this)
- Flowcharts concept page

man or help:

• printf (3)

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 (https://github.com/holbertonschool/Betty/blob/master/betty-style.pl) and betty-doc.pl (https://github.com/holbertonschool/Betty/blob/master/betty-doc.pl)
- · You are not allowed to use global variables
- · No more than 5 functions per file
- 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 should be included in your header file called main.h
- Don't forget to push your header file
- · All your header files should be include guarded
- Note that we will not provide the _putchar function for this project

GitHub

There should be one project repository per group. The other members do not fork or clone the project to ensure only one of the team has the repository in their github account otherwise you risk scoring 0%

More Info

Authorized functions and macros

- write (man 2 write)
- malloc (man 3 malloc)
- free (man 3 free)
- va_start (man 3 va_start)

- va_end (man 3 va_end)
- va_copy (man 3 va_copy)
- va_arg (man 3 va_arg)

Compilation

• Your code will be compiled this way:

```
$ gcc -Wall -Werror -Wextra -pedantic -std=gnu89 *.c
```

- As a consequence, be careful not to push any c file containing a main function in the root directory of your project (you could have a test folder containing all your tests files including main functions)
- Our main files will include your main header file (main.h): #include main.h
- You might want to look at the gcc flag -Wno-format when testing with your _printf and the standard printf. Example of test file that you could use:

```
alex@ubuntu:~/c/printf$ cat main.c
#include <limits.h>
#include <stdio.h>
#include "main.h"
/**
 * main - Entry point
 * Return: Always 0
 */
int main(void)
{
    int len;
    int len2;
    unsigned int ui;
    void *addr;
    len = _printf("Let's try to printf a simple sentence.\n");
    len2 = printf("Let's try to printf a simple sentence.\n");
    ui = (unsigned int)INT_MAX + 1024;
    addr = (void *)0x7ffe637541f0;
    _printf("Length:[%d, %i]\n", len, len);
    printf("Length:[%d, %i]\n", len2, len2);
    _printf("Negative:[%d]\n", -762534);
    printf("Negative:[%d]\n", -762534);
    _printf("Unsigned:[%u]\n", ui);
    printf("Unsigned:[%u]\n", ui);
    _printf("Unsigned octal:[%o]\n", ui);
    printf("Unsigned octal:[%o]\n", ui);
    _printf("Unsigned hexadecimal:[%x, %X]\n", ui, ui);
    printf("Unsigned hexadecimal:[%x, %X]\n", ui, ui);
    _printf("Character:[%c]\n", 'H');
    printf("Character:[%c]\n", 'H');
    _printf("String:[%s]\n", "I am a string !");
    printf("String:[%s]\n", "I am a string !");
    _printf("Address:[%p]\n", addr);
    printf("Address:[%p]\n", addr);
    len = _printf("Percent:[%%]\n");
```

```
len2 = printf("Percent:[%%]\n");
   _printf("Len:[%d]\n", len);
   printf("Len:[%d]\n", len2);
   _printf("Unknown:[%r]\n");
   printf("Unknown:[%r]\n");
   return (0);
}
alex@ubuntu:~/c/printf$ gcc -Wall -Wextra -Werror -pedantic -std=gnu89 -Wno-format *.c
alex@ubuntu:~/c/printf$ ./printf
Let's try to printf a simple sentence.
Let's try to printf a simple sentence.
Length:[39, 39]
Length:[39, 39]
Negative: [-762534]
Negative: [-762534]
Unsigned: [2147484671]
Unsigned: [2147484671]
Unsigned octal:[20000001777]
Unsigned octal:[20000001777]
Unsigned hexadecimal: [800003ff, 800003FF]
Unsigned hexadecimal: [800003ff, 800003FF]
Character:[H]
Character:[H]
String:[I am a string !]
String:[I am a string !]
Address: [0x7ffe637541f0]
Address: [0x7ffe637541f0]
Percent: [%]
Percent: [%]
Len:[12]
Len:[12]
Unknown: [%r]
Unknown: [%r]
alex@ubuntu:~/c/printf$
```

- We strongly encourage you to work all together on a set of tests
- If the task does not specify what to do with an edge case, do the same as printf

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.

Tasks

0. I'm not going anywhere. You can print that wherever you want to. I'm here and I'm a Spur for life

mandatory

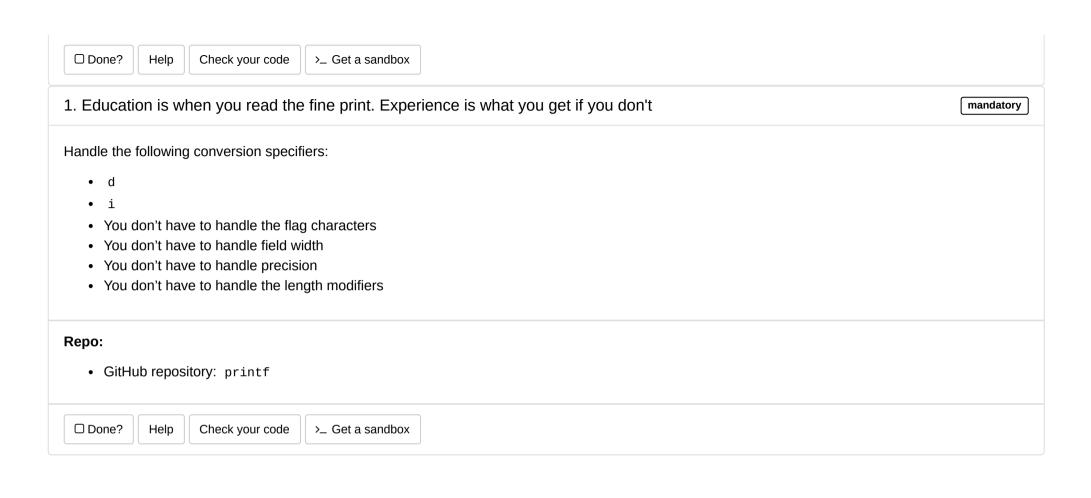
Write a function that produces output according to a format.

- Prototype: int _printf(const char *format, ...);
- Returns: the number of characters printed (excluding the null byte used to end output to strings)
- write output to stdout, the standard output stream
- format is a character string. The format string is composed of zero or more directives. See man 3 printf for more detail. You need to handle the following conversion specifiers:
 - o C
 - o S
 - 0 %
- You don't have to reproduce the buffer handling of the C library printf function
- · You don't have to handle the flag characters
- · You don't have to handle field width
- You don't have to handle precision
- You don't have to handle the length modifiers

Repo:

• GitHub repository: printf

C



2. With a face like mine, I do better in print

#advanced

Handle the following custom conversion specifiers:

• b: the unsigned int argument is converted to binary

```
alex@ubuntu:~/c/printf$ cat main.c
#include "main.h"

/**
 * main - Entry point
 *
 * Return: Always 0
 */
int main(void)
{
    _printf("%b\n", 98);
    return (0);
}
alex@ubuntu:~/c/printf$ gcc -Wall -Wextra -Werror -pedantic -std=gnu89 main.c
alex@ubuntu:~/c/printf$ ./a.out
1100010
alex@ubuntu:~/c/printf$
```

Repo:

• GitHub repository: printf

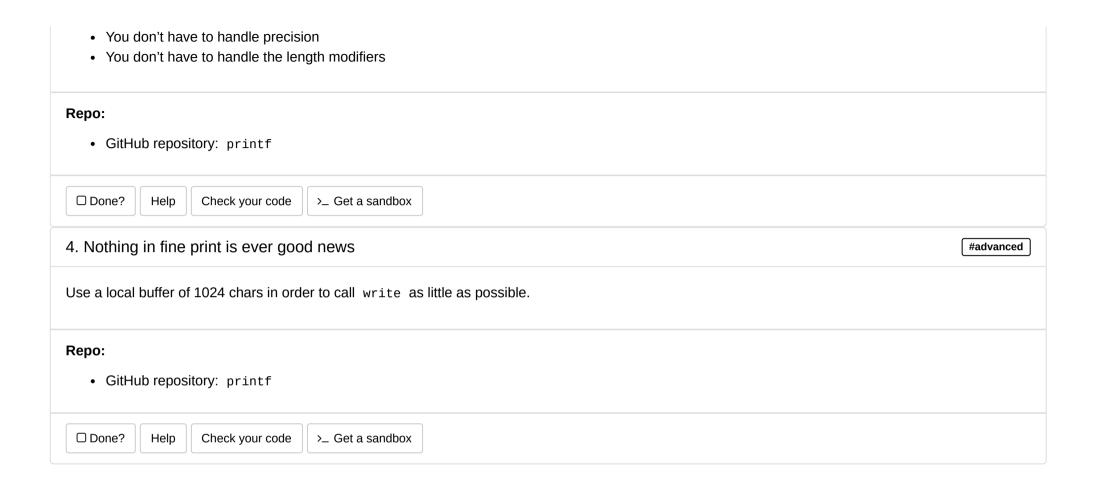
3. What one has not experienced, one will never understand in print

#advanced

Handle the following conversion specifiers:

- u
- 0
- X
- X
- You don't have to handle the flag characters
- You don't have to handle field width

Q



5. My weakness is wearing too much leopard print

#advanced

Handle the following custom conversion specifier:

- S : prints the string.
- Non printable characters (0 < ASCII value < 32 or >= 127) are printed this way: \x , followed by the ASCII code value in hexadecimal (upper case always 2 characters)

Q

```
alex@ubuntu:-/c/printf$ cat main.c
#include "main.h"

/**
  * main - Entry point
  *
  * Return: Always 0
  */
  int main(void)
{
    _printf("%S\n", "Best\nSchool");
    return (0);
}
alex@ubuntu:-/c/printf$ gcc -Wall -Wextra -Werror -pedantic -std=gnu89 main.c
alex@ubuntu:-/c/printf$ ./a.out
Best\x0ASchool
alex@ubuntu:-/c/printf$
```

Repo:

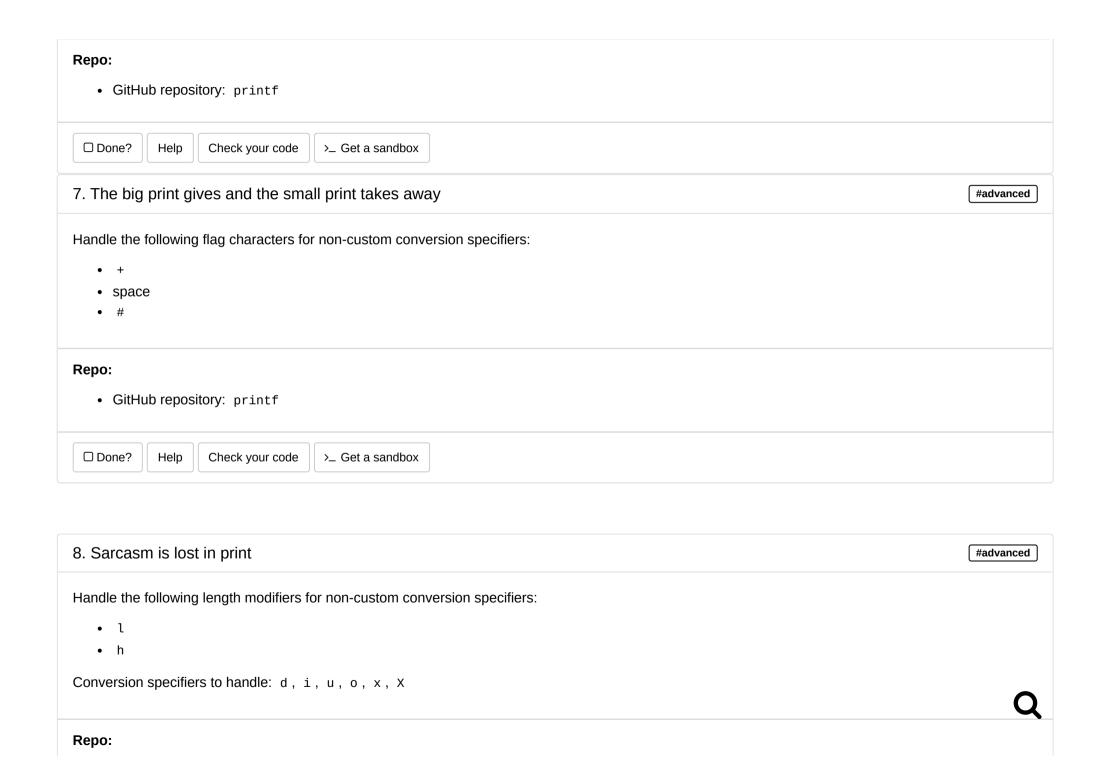
• GitHub repository: printf

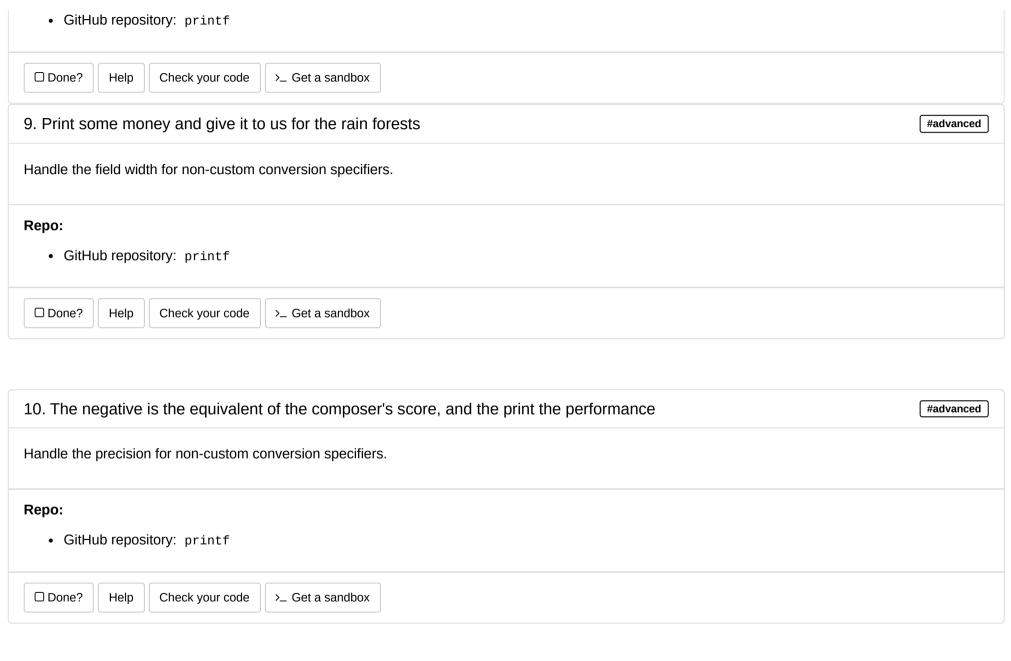
6. How is the world ruled and led to war? Diplomats lie to journalists and believe these lies when they see them in print

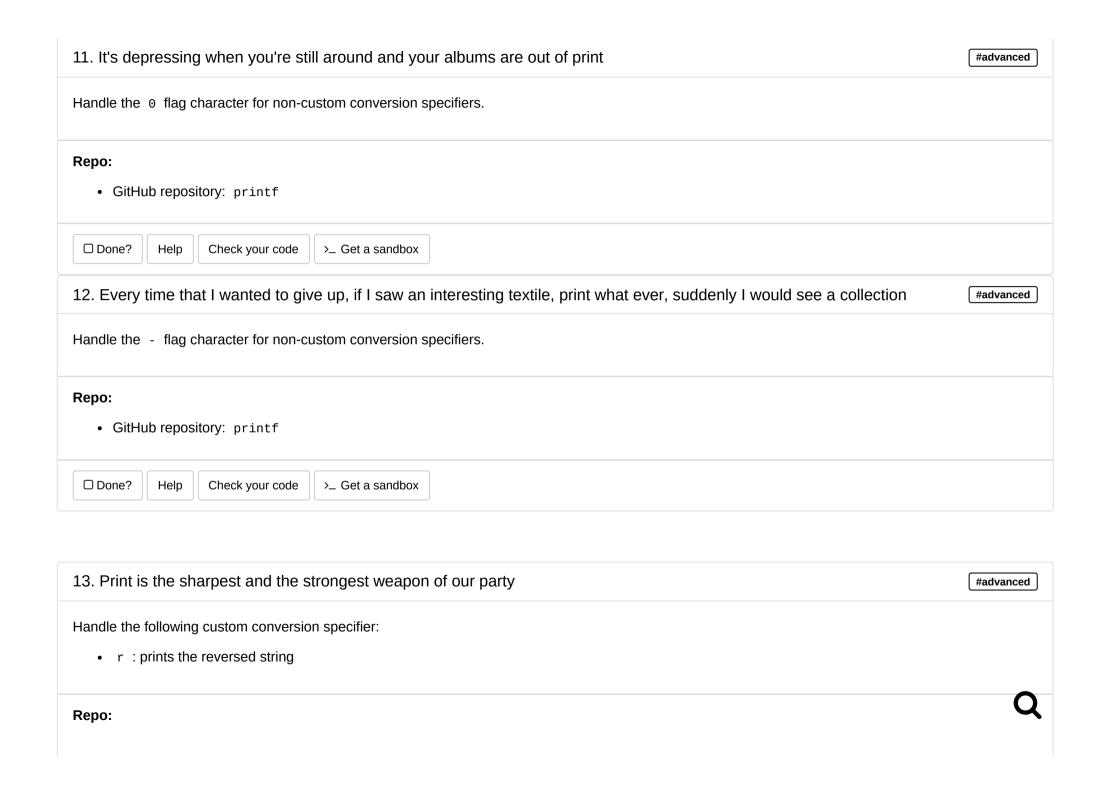
#advanced

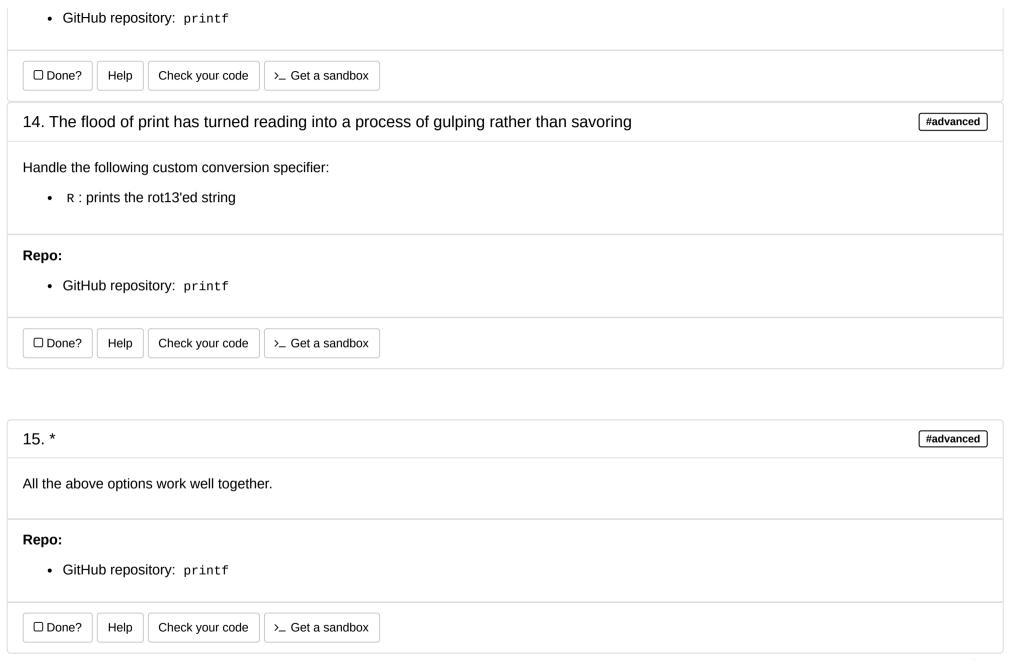
Handle the following conversion specifier: $\, p \, . \,$

- You don't have to handle the flag characters
- You don't have to handle field width
- You don't have to handle precision
- You don't have to handle the length modifiers









Copyright © 2023 ALX, All rights reserved.