FT_PRINTF int ft_printf(char * formato, arg1,...,argn)			**Note**
La funzione printf() scrive su stdout gli argomenti passati, eseguendo le opportune conversioni sotto il controllo della stringa di **formattazione format.** I caratteri ordinari contenuti in format vengono copiati direttamente sullo stdout, mentre le **direttive di conversione** specificano come devono essere trattati gli argomenti che seguono **Sequenze di escape** Il carattere \ (backslash) non viene trasferito sullo schermo, ma utilizzato in combinazione con i caratteri successivi (un solo carattere se si tratta di una lettera, oppure una sequenza di cifre numeriche); l'insieme viene detto: escape sequence, e viene interpretato come un unico carattere. Le sequenze di escape sono usate tipicamente per specificare caratteri speciali che non hanno il loro equivalente stampabile (come newline, carriage return, tabulazioni, suoni ecc...), oppure caratteri, che da soli, hanno una funzione speciale, come le virgolette o lo stesso backslash Ogni direttiva di conversione inizia con il carattere % e termina con uno dei caratteri **d, i, o, x, X, u, c, s, f, e, E, g, E, p, n, %.** Tra il **carattere di inizio %** e il carattere di conversione possono essere specificati dei **flag** (**in ordine qualsiasi**), l'ampiezza, la precisione ed un modificatore.			se non ci sono argomenti l'output e' una stringa nulla(per le stringhe) o uno zero (per i numeri)
Struttura della direttiva di conversione **ft_printf("questo e il mio int %050.6d e questo e il mio char %c", int, char)** **%[flags][width][.precision][length]specifier** **se la precisione e maggiore dell'ampiezza e l'ampiezza e c'e la flag 0, l ampiezza cambia la flag da zero a spazio.**			
PARTE BONUS: A SCELTA TRA Manage any combination of the following flags: '-0.' and the field minimum width under all conversions. Manage all the following flags: '# +' (Yes, one of them is a space). **NB: Per il 25% di bonus è necessario gestire tutte e 6 le flag**			
%	Obbligatorio	Indica l'inizio di una direttiva di conversione.	
-	**Flag opzionale**	**Allinea la scrittura a sinistra in un campo con ampiezza specificata da width. Specifica un incolonnamento a sinistra.** Il valore convertito viene riempito a destra con spazi, piuttosto che a sinistra con spazi vuoti o zeri. A - sostituisce uno 0 se sono dati entrambi.	con le stringhe stampa nspazi meno la lunghezza della stringa
0	**Flag opzionale**	Aggiunge zeri sulla sinistra fino a raggiungere l'ampiezza specificata da width. Se appaiono insieme 0 e -, 0 è ignorato.Se compaiono entrambi i flag 0 e -, il flag 0 viene ignorato. **Se viene fornita una precisione con una conversione numerica (d, i, o, u, x e X), il flag 0 viene ignorato.** Per altre conversioni, il	solo su d,i,u, x,X. NB: ignorata con la precisione

		risultato è indefinito.	
'.' (punto)	Opzionale	In uno specificatore di formato il campo opzionale precision, se presente, deve essere sempre preceduto da un punto (che lo separa dal campo width), ed è costituito da un numero intero non negativo, con significato che dipende dal contenuto del campo obbligatorio type, come si evince dal campo precisione piu avanti. Se viene indicato il punto, ma viene omessa la precisione, questa e' assunta a zero.	
+	**Flag opzionale**	Specifica che il **segno** verrà sempre stampato. In genere il segno è usato solo per i numeri negativi, il + sovrascrive lo spazio se sono usati entrambi.	di default Il segno appare solo se il numero è negativo
' ' (spazio)	**Flag opzionale**	Specifica che verrà stampato uno spazio prima di un numero positivo **.**	solo su d,i,u, x,X
#	**Flag opzionale**	%#x (Hex) 0x prefix added to non-zero values %#X (Hex) 0X prefix added to non-zero values. Per altre conversioni, il risultato è indefinito.	
Ampiezza	**Numero opzionale**	In uno specificatore di formato il campo opzionale width, costituito da un numero intero positivo, determina l'ampiezza di campo, cioè il numero di caratteri minimo con cui deve essere scritto il dato corrispondente. Se il numero di caratteri effettivo è inferiore, il campo viene riempito (normalmente) a sinistra con spazi bianchi; se invece il numero è superiore, il campo viene espanso fino a raggiungere la lunghezza effettiva (in altre parole il dato viene sempre scritto per intero, anche se il valore specificato in width è insufficiente). ~~Se al posto di un numero si specifica nel campo width un asterisco, il valore viene desunto in esecuzione dalla lista degli argomenti della printf; in questo caso il valore dell'ampiezza di campo deve precedere immediatamente il dato a cui lo specificatore in esame si riferisce.~~	
Precisione	**Numero opzionale**	**Per d,i,u,o,x,X:** La precisione specifica il minimo numero di cifre che devono essere scritte. Se il numero di cifre effettive del dato corrispondente è minore della precisione, vengono scritti degli zeri sulla sinistra fino a completare il campo. Se invece il numero di cifre effettive è superiore, il dato è comunque scritto per intero senza nessun troncamento. Infine, se la precisione è .0 (oppure semplicemente .) e il dato è zero, non viene scritto nulla. default 1 **Per c:** La precisione non ha effetto.	

		Per s: La precisione specifica il massimo numero di caratteri che devono essere scritti. I caratteri in eccesso non vengono scritti. non aggiunge gli zeri. defoult: tutta la stringa **Per p x e X:** se e maggiore mi stampa il ptr se e minore mi aggiunge gli 0. ~~La precisione puo' essere sostituita dal carattere *. In tal caso il valore della precisione viene letto dal prossimo argomento che deve essere di tipo **int**.~~	
~~**Modificatore**~~ ~~**h, l** e ~~**L**~~	~~**Lettera opzionale**~~	~~La lettera **h** seguita da uno dei caratteri di conversione **d**, **i**, **u**, **o**, **x** o **X** indica che l'argomento e' uno **short int** o uno **unsigned short int**.~~ ~~Il modificatore **l** (elle) con gli stessi caratteri di conversione, indica che l'argomento e' un **long int** o uno **unsigned long int**.~~ ~~Il modificatore **L** seguito da uno dei caratteri di conversione **e**, **E**, **f**, **g** o **G** indica che l'argomento e' un **long double**.~~	

Caratteri di conversione (Lettera Obbligatoria) (ONLY cspdiuxX%)

CARATTERE	TIPO DI ARG	SPECIFICA DI CONVERSIONE	NOTE
d, **i**	**int**	L'argomento e' convertito a **decimale segnato**	
u	**int**	L'argomento e' convertito in **decimale senza segno**.	
~~**o**~~	~~**int**~~	~~L'argomento e' convertito in **notazione ottale** e **priva di segno**~~	Non previsto
x, **X**	**int**	L'argomento e' convertito **in notazione esadecimale priva di segno.** A secondo che sia specificato **x** o **X**, le cifre esadecimali impiegate sono rispettivamente **abcdef** o **ABCDEF**. Nella tecnologia informatica, una cifra o posizione binaria corrisponde a un bit. Un byte comprende 8 bit e un mezzo byte, chiamato anche nibble, è formato da 4 bit. Questo significa che una cifra esadecimale può essere usata per rappresentare un **nibble** e due cifre esadecimali possono essere usate per rappresentare **un byte completo**	

c	**int**	L'argomento e' convertito in **unsigned char** e viene scritto il carattere risultante.	
s	**char ***	Stampa la **stringa** individuata dall'argomento, fino a raggiungere il carattere '\0'.	
f	**double**	Converte l'argomento **in notazione decimale** nel formato [-]ddd.ddd dove le cifre della parte decimale sono date dalla precisione, che per default vale 6.	Non previsto
e, E	**double**	L'argomento e' stampato **in notazione esponenziale**, in cui viene riportata la mantissa (parte decimale di un numero reale ottenuta facendo la differenza tra il numero e la sua parte intera), la parte decimale e l'esponete preceduto da **e** oppure da **E**, in funzione del carattere di conversione indicato. La parte decimale e' determinata dalla precisione (default = 6).	Non previsto
g, G	**double**	L'argomento e' convertito con le regole definite per il carattere di conversione **f** o **e** o **E**. **e** o **E** viene utilizzato se l'esponente e' < -4 o e' maggiore o uguale alla precisione. Viene utilizzato **f** nell'altro caso.	Non previsto
p	**void ***	**stampa** in formato esadecimale!!!! Viene stampato il valore del **puntatore** passato come argomento.	
n	**int ***	Il numero di caratteri scritti vengono **memorizzati nell'intero puntato dall'argomento.** **Nessun argomento viene convertito**.	Non previsto
%		Stampa il carattere '%'. **Nessun argomento viene trattato.**	

SUBJECT **int ft_printf(const char *, ...);**	
Don’t implement the buffer management of the original printf().	Gli output di un programma vengono memorizzati in un buffer e quando si chiama printf vengono estratti. il metodo **setbuf()** viene utilizzato per gestire quel buffer. Quando aggiungi **setbuf(stdout , NULL),** questo dice al buffer che non memorizza gli output e invia direttamente gli output del programma nello stdout.
Your function has to handle the following conversions: cspdiuxX%	VEDI CARATTERI DI CONVERSIONE
You must use the command ar to create your library. Using the libtool command is forbidden.	
Your libftprintf.a has to be created at the root of your repository	

Your functions should not quit unexpectedly (segmentation fault, bus error, double free, etc) apart from undefined behaviors.	
All heap allocated memory space must be properly freed when necessary. No leaks will be tolerated.	
If your project allows you to use your libft, you must copy its sources and its associated Makefile in a libft folder with its associated Makefile. Your project's Makefile must compile the library by using its Makefile, then compile the project.	

PROGETTO		
1	COMPILARE LA LIBRERIA	`#include "ft_printf.h"` `#include "ft_libft.h"`
2	**COSTRUIRE DELLE STRUTTURE PER LE FLAG E LE VARIABILI**	in base alla variabile richiamo la flag
3	**IMPLEMENTARE FUNZIONI DI COPIA E DI SCRITTURA**	va_list?
4	**IMPLEMENTARE PRINTF**	
	Tester:	https://github.com/Tripouille/printfTester

FILES	
Program name	libftprintf.a
Turn in files	Makefile, *.h, */*.h, *.c, */*.c
Makefile	NAME, all, clean, fclean, re
External functs.	malloc, free, write, va_start, va_arg, va_copy, va_end
Libft authorized	Yes
Description	Write a library that contains ft_printf(), a function that will mimic the original printf()

va_list	
va_start void va_start(va_list arg_ptr, prev_param);	imposta arg_ptr sul primo argomento **facoltativo** nell'elenco di argomenti passati alla funzione. L'argomento arg_ptr deve essere di tipo va_list. Prev_param è il nome del parametro obbligatorio che precede il primo argomento facoltativo nell'elenco di argomenti. Se prev_param viene

	dichiarata con la classe di archiviazione nel registro, il comportamento della macro non è definito. È necessario usare va_start prima di usare va_arg per la prima volta.

va_arg
 	
type va_arg(
  va_list
arg_ptr,
  type
);
	recupera un valore di type dal percorso specificato da arg_ptr e incrementa arg_ptr in modo che punti all'argomento successivo nell'elenco usando le dimensioni di type per determinare dove inizia l'argomento successivo. È possibile usare va_arg un qualsiasi numero di volte nella funzione per recuperare gli argomenti dall'elenco.

va_copy
 	
void va_copy(
  va_list dest,
  va_list src
);
	crea una copia di un elenco di argomenti nello stato corrente. Il parametro src deve essere già stato inizializzato con va_start. Potrebbe essere stato aggiornato con chiamate a va_arg, ma non deve essere stato reimpostato con va_end. L'argomento successivo che viene recuperato da va_arg da dest corrisponde all'argomento successivo recuperato da src.

va_end void va_end(va_list arg_ptr);	reimposta il puntatore su NULL. È necessario chiamare va_end per ogni elenco di argomenti inizializzato con va_start o va_copy prima che la funzione restituisca il controllo.

```
c = va_arg(args, int)
s = va_arg(args, char *)
d = va_arg(args, int)
i = va_arg(args, int)
u = va_arg(args, unsigned int)
p = va_arg(args, unsigned long)# or
                        #(unsigned long)va_arg(args, void *);
x = va_arg(args, unsigned int)
X = va_arg(args, unsigned int)
```