**Program name :** libftprintf.a

**Turn in files :** \*.c, \*/\*.c, \*.h, \*/\*.h, Makefile

**Makefile :** all, clean, fclean, re, bonus

**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 real printf

printf를 흉내내는 ft\_printf가 포함딘 라이브러리를 작성해라.

**mandatory part**

* prototype : int ft\_printf(const char \*, …);
* 실제 printf처럼 buffer management를 하면 안됨.
* 다음의 conversion을 manage할 것이다 : cspdiuxX%
* 다음의 flag의 조합으로 manage할 것이다 : ‘-0.\*’ 그리고 최소 field width의 모든 conversion
* man 3 printf / man 3 stdarg

**man 3 printf**

int ft\_printf(const char \*format, …);

The format string is a character string, beginning and ending in its initial shift state, if any. The format string is composed of zero or more directives: ordinary characters (not %), which are copied unchanged to the output stream; and conversion specifications, each of which results in fetching zero or more subsequent arguments. Each conversion specification is introduced by the character %, and ends with a conversion specifier. In between there may be (in this order) zero or more flags, an optional minimum field width, an optional precision and an optional length modifier.

format string은 문자열이며, 초기 이동 상태에서 시작 및 종료된다. format 문자열은 0개 이상의 지시문(즉 output stream에 변경되지 않고 복사되는 일반 문자(%아님))과 conversion specification 으로 구성되며, 각각 0개 이상의 subsequent argument를 가져오게 된다.(result in fetching). 각 conversion specification은 %문자로 소개되며 conversion specifier로 끝난다. 그 사이에(이 순서에 따라) 0개 이상의 flag들, 선택적인 minimum field width, 선택적인 precision(정밀도), 선택적인 length modifier가 있을 수 있다.

The arguments must correspond properly (after type promotion) with the conversion specifier. By default, the arguments are used in the order given, where each '\*' (see Field width and Precision below) and each conversion specifier asks for the next argument (and it is an error if insufficiently many arguments are given). One can also specify explicitly which argument is taken, at each place where an argument is required, by writing "%m$" instead of '%' and "\*m$" instead of '\*', where the decimal integer m denotes the position in the argument list of the desired argument, indexed starting from 1.

argument는 (type promotion 후에) conversion specifier과 적절하게 일치해야 한다. 기본적으로, argument는 주어진 순서에 따라 사용되며, 각 ‘\*’(아래의 Field width와 Percision을 보라)과 각 conversion specifier는 next argument를 요청한다(그리고 불충분한 많은 argument가 주어진 경우 오류). 또한 argument가 필요한 위치에서 “%m$” 대신 ‘%’와 “\*m$” 대신 ‘\*’를 작성함으로써 argument가 필요한 각 지점에서 decimal integer m이 1부터 시작하는 argument의 argument list의 position을 나타냄으로써 어떤 argument를 취할 것인지를 명시적으로 지정할 수 있다.

Thus,

printf("%\*d", width, num);

and

printf("%2$\*1$d", width, num);

are equivalent. 둘이 같다.

The second style allows repeated references to the same argument. The C99 standard does not include the style using '$', which comes from the Single UNIX Specification. If the style using '$' is used, it must be used throughout for all conversions taking an argument and all width and precision arguments, but it may be mixed with "%%" formats, which do not consume an argument. There may be no gaps in the numbers of arguments specified using '$'; for example, if arguments 1 and 3 are specified, argument 2 must also be specified somewhere in the format string.

두번째 스타일은 같은 argument에 대한 반복적인 reference(언급)을 허용한다. C99 standard에는 single UNIX Specification에서 나온 ‘$’를 사용한 스타일은 포함되지 않는다. ‘$’를 사용하는 경우, argument를 사용하는 모든 conversion과 모든 width과 precision argument에 대해 전체적으로 사용해야 하지만 argument를 사용하지 않는 “%” 형식과 믹스할 수 있다. 예를들어, argument 1과 3이 지정된 경우, argument 2도 format string 어딘가에 지정해야한다.(specify)

For some numeric conversions a radix character ("decimal point") or thousands' grouping character is used. The actual character used depends on the LC\_NUMERIC part of the locale. (See setlocale(3).) The POSIX locale uses '.' as radix character, and

does not have a grouping character. Thus,

printf("%'.2f", 1234567.89);

results in "1234567.89" in the POSIX locale, in "1234567,89" in the nl\_NL locale, and in "1.234.567,89" in the da\_DK locale.

**field width**

An optional decimal digit string (with nonzero first digit) specifying a minimum field width. If the converted value has fewer characters than the field width, it will be padded with spaces on the left (or right, if the left-adjustment flag has been given). Instead of a decimal digit string one may write "\*" or "\*m$" (for some decimal integer m) to specify that the field width is given in the next argument, or in the m-th argument, respectively, which must be of type int. A negative field width is taken as a '-' flag followed by a positive field width. In no case does a nonexistent or small field width cause truncation of a field; if the result of a conversion is wider than the field width, the field is expanded to contain the conversion result.

minimum field width를 지정하는 선택적인 decimal digit string(first digit이 0이 아님). 변환된 값에 field width보다 작은 문자열이 있는 경우, 왼쪽에 공백이 pading된다.

field width : printf(“%10.6f”)

에서 10이 field width임.

precision : . 이후의 숫자는 실제로 출력될 글자의 최소 길이. 글자수가 precision의 길이보다 작으면 0으로 채움.

ex) printf("%.2x", 1u) 하면 "01"이 출력된다는 뜻이지요.

* %d : signed 10진수 정수를 입력 받음
* %i : 10진수/8진수/16진수 입력 받음

**bonus part**

* 최소 한 개 이상의 다음 conversion을 manage : nfge
* 최소 한 개 이상의 다음 flag를 manage : l ll h hh
* 다음의 모든 flag를 manage : ‘# +’ (yes, 이 중 하나는 space임).