MyConv

Deadline: September 12th, 2014

1 Instructions

You must implement the following functions:

```
int my_dec2int(const char *s);
unsigned int my_dec2uint(const char *s);
unsigned int my_hex2uint(const char *s);
int my_int2dec(char *dst, int v, unsigned n);
int my_uint2dec(char *dst, unsigned v, unsigned n);
int my_uint2hex(char *dst, unsigned v);
```

You can optionally also implement the following for a higher grade:

```
long my_strtol(const char *str, char **endptr, int base);
unsigned long my_strtoul(const char *str, char **endptr, int base);
```

- each function must be implemented in a `.c` file of its own, named after the function it contains. The function prototypes must be declared in a `.h` file, in accordance with the C coding standard. The submitted archive may (but needs not) include a test program.
- you must not include any standard/system header in your code; nor use any function from the standard C library. You may use functions from a previous assignment (by including their source in your submission).

2 Function semantics

`my_dec2int(x)`, `my_dec2uint(x)` and `my_hex2uint(x)` all read the representation of a number stored in the nul-terminated string `x` and returns it as a C integer. For `my_dec2int`, the input string may start with a minus sign if the number is negative; for all three the remainder of the input string are valid digits. As the name implies, `my_dec2int` and `my_dec2uint` read a decimal number representation, whereas `my_hex2uint` reads an hexadecimal number representation (with digits 0-9 or A-F in either upper or lower case).

The first 3 functions will be tested mainly with operands that are guaranteed to fit their output type. Optionally, you can choose to implement saturation for exceedingly large operands, ie. round to the closest C integer.

`my_int2dec(d, v, n)` places the decimal representation of `v` into the buffer pointed to by `d`, using up to a maximum of `n` characters, and returns the number of characters actually written.

`my_uint2dec` does the same starting with an unsigned integer; and `my_uint2hex` does the same using base 16. If the output buffer is not large enough, then the conversion must not take place and the function must return 0.

The optional functions `my_strtol` and `my_strtoul` must match the documentation of the standard C functions of the same names (without the `my_` prefix).

3 Grading

- 1 point per function correctly implemented in the mandatory list.
- +0.5 if all of the above, and a `Makefile` places the functions in `libminic.a`.
- +1 per optional function correctly implemented after all of the above.
- +0.5 if saturation is properly implemented.
- +1 if the code is properly factored, ie. each conversion direction reuses the same code from multiple functions.

Beware of edge cases! Test your functions very carefully.

---

4 Copyright and licensing