

```

/*
Chau Nguyen
tug37553@temple.edu
lab Float
*/

#include <stdio.h>
#include <limits.h>
#include <math.h>
#include <float.h>
#include <stdlib.h>

#define NORM 0
#define DNORM 1
#define SPEC 2
#define BIAS 127

//declare struct
typedef struct{
    int sign;
    int exp;
    float man;
    int mode;
}flt;

//prototype
int get_flt_bits_int(float f);
char get_flt_sign_char(float f);
int get_flt_sign_val(float f);
char * get_flt_exp_str(float f);
int get_flt_exp_val(float f);
char get_flt_exp_mode(float f);
char * get_flt_man_str(float f);
float get_flt_man_val(float f);
flt get_flt_val_flt(flt bits, float f);
void print_flt(flt bits);
char * get_flt_bits_str(float f);
float get_flt_bits_va(flt bits);
/*
    Write a function get_flt_bits_int to return an integer with the
    bits copied from a float.
    Example:
        for f = -15.375,
        the bits of int n = 11000001011101100000000000000000
    Look at the slides and code from the float lectures and use the
    indirection trick. This can easily be done in one line of code.
    The function should accept a float and return an int.
*/
int get_flt_bits_int(float f){
    int i =*(int*)&f; //int is presentation the bits
    return i;
}

/*
    Write a function that returns the sign of a float as a char.
    You should call get_flt_bits_int to get the bits in an int
    and return '1' if the sign is negative else return '0'. The
    function should accept a float and return a string.
*/
char get_flt_sign_char(float f){
    char sign;
    if(get_flt_bits_int(f)&INT_MIN){
        sign='1';
    }else sign='0';
    return sign;
}

```

```

    }

/*
Write a function that returns the sign of a float as an integer.
You should call get_flt_bits_int to get the bits in an int
and return -1 if the sign is negative else return 1. The function
should accept a float and return an int.
*/
int get_flt_sign_val(float f){
    int sign;
    if(get_flt_bits_int(f)&INT_MIN){
        sign=-1;
    }else sign= 1;
    return sign;
}

/*
Write a function to return a string containing the
actual binary value of the exponent of a float in a
char array. You should call get_flt_bits_int to get
the bits in an int and return the string.
Example:
    for f = -15.375
        n = 11000001011101100000000000000000
        n = 11000001010101100000000000000000
        the exponent bits are "10000010"
The function should accept a float and return a string.
*/
//return string
char * get_flt_exp_str(float f){
    char * exp = (char *)malloc(9);
    int size=0;
    exp[8]='\0';
    for (int i=(sizeof(int)*8-2);i>=23;i--){
        int n = get_flt_bits_int(f)>>i&1;
        exp[size++] = (char)(n + '0');
    }
    return exp;
}

/*
Write a function to return an integer containing the
actual integer value of the exponent of a float. You
should call get_flt_bits_int to get the bits in an int
and return the int with the exponent value.
Example:
    for f = -15.375
        n = 11000001011101100000000000000000
        the exponent bits are 10000010
        the actual value of the exponent is 3
The function should accept a float and return an int.
*/
int get_flt_exp_val(float f){
    int sum=0;
    int power=7;
    for (int i=(sizeof(int)*8-2);i>=23;i--){
        int n = get_flt_bits_int(f)>>i&1;
        sum+= n*pow(2,power--);
    }
    if(sum==0){

```

```

        return 0;
    }else if(sum==255){
        return 255;
    }else return sum-BIAS;
}

/*
Write a function to return an integer containing the
mode of the exponent of a float. You should call
get_flt_exp_val to get the bits in an int and return
the int with the mode value.
Example:
    for f = -15.375
        n = 11000001011101100000000000000000
        the exponent bits are 10000010
        the mode is NORM
The function should accept a float and return an int.
*/

char get_flt_exp_mode(float f){
    int mode;
    int exp= get_flt_exp_val(f);
    if(exp==0){
        //Denormalized exp is all zero
        mode=DNORM;
    }else if(exp==255){
        //Special exp is all 1
        mode=SPEC;
    }else mode=NORM;
    return mode;
}

/*
Write a function to return a string containing the
actual binary value of the mantissa of a float in a
char array. You should call get_flt_bits_int to get
the bits in an int and return the string.
Example:
    for f = -15.375
        n = 11000001011101100000000000000000
        the mantissa bits are "1110110000000000000000"
The function should accept a float and return a string.
*/

char* get_flt_man_str(float f){
    char * man= (char *)malloc(24);
    int size=0;
    man[23]='\0';
    int n;
    for (int i=(sizeof(int)*8-10);i>=0;i--){
        n = get_flt_bits_int(f)>>i&1;
        man[size++] = (char)(n + '0');
    }
    return man;
}

/*
Write a function to return a float containing the
actual float value of the mantissa of a float. You
should call get_flt_bits_int to get the bits in an int
and return the int with the mantissa value.
Example:

```

```

    for f = -15.375
        n = 11000001011101100000000000000000
        the mantissa bits are 111011000000000000000000
        the actual value of the mantissa is 0.9218750000
The function should accept a float and return an int.
*/

float get_flt_man_val(float f){
    float man=0;
    int bits;
    int power=1;
    for (int i=(sizeof(int)*8-10);i>=0;i--){
        bits = get_flt_bits_int(f)>>i&1;
        int pow1=pow(2,power++);
        float new= ((float)bits/(float)pow1);
        man+=new;
    }
    return man;
}

/*
Write a function to return a string containing the
actual binary value of a float in a char array. You
should call get_flt_sign_char, get_flt_exp_str and
get_flt_man_str to get the bits in an char and two
strings and return the concatenated string.
Example:
    for f = -15.375
        n = 11000001011101100000000000000000
        The sign is '1'
        the exponent is "10000010"
        and the mantissa bits are "111011000000000000000000"
        The string should be formatted as:
            "1 10000010 111011000000000000000000" to clearly
            separate the 3 parts.
The function should accept a float and return a string.
*/
char * get_flt_bits_str(float f){
    int j=0;
    char * str = (char *)malloc(sizeof(str)*35);
    char sign =get_flt_sign_char(f);
    char* exp =get_flt_exp_str(f);
    char * man= get_flt_man_str(f);
    //add sign char
    str[34]='\n';
    str[j++]=sign;
    str[j++]=' ';
    for(int i=0; exp[i]!= '\0'; i++,j++){
        str[j]=exp[i];
    }
    str[j++]=' ';
    for(int i=0; man[i]!= '\0';i++, j++){
        str[j]=man[i];
    }
    free(exp);
    free(man);
    return str;
}

/*
Write a function to separate the parts of a float

```

```
Example after processing: -15.375 = 1 10000010 1110110000000000000000
  sign = -1
  exp = 3
  man = 0.9218750000
  mode = NORM
```

```
f1t get_flt_val_flt(f1t bits, float f){
    bits.sign=get_flt_sign_val(f);
    bits.exp=get_flt_exp_val(f);
    bits.man=get_flt_man_val(f);
    bits.mode=get_flt_exp_mode(f);
    return bits;
}
```

Write a function to print a `flt` struct to screen. It should accept a `flt` struct and return nothing. Hint: Use `if` statement to print mode.

```
void print_flt(flt bits){
    printf("Output\n");
    printf("Sign = %d\n",bits.sign);
    printf("Exp = %d\n",bits.exp);
    printf("Man = %3f\n",bits.man);
    if(bits.mode==DNORM){
        printf("Mode = de-normalized\n");
    }else if(bits.mode==SPEC){
        printf("Mode = specialized\n");
    }else printf("Mode = normalized\n");
}
```

The float value produced will depend on the mode.
To set a float to infinity use the math library constant INFINITY
To set a float to not-a-number use the math library constant NAN
Check the slides and text for conditions for NORN, DNORM and SPEC
You need to return (sign) * M * 2^e

```
float get_flt_bits_va(flt bits){
    float f;
    if(bits.mode==DNORM){
        //denormalized exp=0
        f=(float)bits.sign*NAN;
    }else if(bits.mode==SPEC){
        //special exp=255
        f=(float)bits.sign*INFINITY;
    }else
        //normalized
        f =(float)bits.sign*(bits.man+1)*(float)pow(2,bits.exp);
    return f;
}
```

Write a main function that calls and prints results for each function when completed.

[illegible]

```

//float f=sqrt(-1);
//float f =-INFINITY;
float f =-15.375;
printf("f = %f\n\n",f);
printf("sign = %c\n",get_flt_sign_char(f));
printf("s = %d\n\n",get_flt_sign_val(f));
//char* exp =get_flt_exp_str(f);
printf("EXP = %s\n",get_flt_exp_str(f));
// int e =get_flt_exp_val(f);
printf("e = %d\n\n", get_flt_exp_val(f));
//char * man= get_flt_man_str(f);
printf("Man = %s\n", get_flt_man_str(f));
// float m =get_flt_man_val(f);
printf("M value is %.9f\n\n",get_flt_man_val(f));
char *bits = get_flt_bits_str(f);
printf("After processing: %f = %s\n",f,bits);
//creat a type flt
flt floatStruct;
floatStruct=get_flt_val_flt(floatStruct,f);
print_flt(floatStruct);
float returnfloat=get_flt_bits_va(floatStruct);
printf("ff = %.9f\n",returnfloat);
free(bits);
}

```