**Problem 1**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Mode | x | | y | | x\*y, w = 10 | | Truncated x\*y, w = 5 | |
| a | Unsigned | 16 | 10000 | 21 | 10101 | 336 | 0101010000 | 16 | 10000 |
| Two’s complement | -16 | 10000 | -11 | 10101 | 176 | 0010110000 | -16 | 10000 |
| b | Unsigned | 21 | 10101 | 8 | 01000 | 168 | 0010101000 | 8 | 01000 |
| Two’s complement | -11 | 10101 | 8 | 01000 | -88 | 1110101000 | 8 | 01000 |
| c | Unsigned | 12 | 01100 | 25 | 11001 | 300 | 0100101100 | 12 | 01100 |
| Two’s complement | 12 | 01100 | -7 | 11001 | -84 | 1110101100 | 12 | 01100 |
| d | Unsigned | 10 | 01010 | 5 | 00101 | 50 | 0000110010 | 18 | 10010 |
| Two’s complement | 10 | 01010 | 5 | 00101 | 50 | 0000110010 | -14 | 10010 |

**Problem 2**

1. K = 17x = 16x + x = x24 + x = **(x << 4) + x**
2. K = -7x = x – 8x = x – x23 = **x – (x << 3)**
3. K = 60x = 64x – 4x = x26 – x22 = **(x << 6) – (x << 2)**
4. K = -112x = 16x – 128x = x24 – x27 = **(x << 4) – (x << 7)**

**Problem 3**

Code –

/\* Print a float in binary: ftob.c \*/

#include <stdio.h>

#include <stdlib.h>

//void float\_to\_string(float f, char \*s, int n);

void float\_to\_string(float,char \*,int);

void print\_float();

#define LEN 32

#define EXP\_32 8 /\* ending index of s for exponent \*/

#define MAN\_32 9 /\* starting index of s for significand \*/

int main(int argc, char \*\*argv) {

int n=LEN;

float f;

char s[LEN];

f = atof(argv[1]);

printf("f=%f\n",f);

float\_to\_string(f,s,n);

print\_float(s,n);

return 0;

}

/\* convert float to binary and store in s, a string of 32 chars \*/

void float\_to\_string(float f, char \*s, int n){

unsigned int u\_int;

int i; /\* for loop index \*/

/\* fill here \*/

unsigned int \*ptr = (unsigned int \*)&f;

u\_int = \*ptr;

int bit;

int char\_index = 0;

for(i = n - 1; i >= 0; i--){

bit = (u\_int >> i) & 1;

s[char\_index++] = (bit == 1) ? '1' : '0';

}

}

/\* print space in between sign bit, exponent, and significand \*/

void print\_float(char \*s, int n) {

int i=0;

/\* fill here \*/

printf("%c ", s[i++]);

for(i = 1; i <= EXP\_32; i++) printf("%c", s[i]);

printf(" ");

for(i = MAN\_32; i < n; i++) printf("%c", s[i]);

printf("\n");

}

/\* End of ftob.c \*/

Example –

A computer screen with white text

AI-generated content may be incorrect.

**Problem 4**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Value** | **Binary** | **Rounded** | **Action** | **Rounded Value** |
| 1 1/16 | 1.0001 | 1.00 | < ½ , down | 1 |
| 1 3/16 | 1.0011 | 1.01 | > ½ , up | 1 ¼ |
| 2 5/16 | 10.0101 | 10.01 | < ½, down | 2 ¼ |
| 2 5/8 | 10.101 | 10.10 | = ½ , even (down) | 2 ½ |
| 3 5/8 | 11.101 | 11.10 | = ½ , even (down) | 3 ½ |
| 3 7/8 | 11.111 | 100.00 | = ½, even (up) | 4 |

**Problem 5**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Value** | **Rounded** | **Exp** | **Adjusted** | **Result** |
| 256 | 1.0000 | 8 |  | Infinity |
| 31 | 1.1111 | 4 |  | 31 |
| 33 | 1.0000 | 5 |  | 32 |
| 35 | 1.0010 | 5 |  | 36 |
| 276 | 1.0001 | 8 |  | NaN |
| 127 | 10.0000 | 6 | 1.0000/7 | 128 |

**Problem 6**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

// Usage -> $ ./p6 0 01111110 10000000000000000000000

int main(int argc, char \*argv[]){

// v=(–1)^s \* M \* 2^E n: E=Exp–Bias(127)

int sign, exponent = 0;

float mantissa;

int bias = 127;

// sign var 0 or 1

sign = atoi(argv[1]);

int val = pow(2, 7);

// calculate exponent value E = exp - bias

for(int i = 0; i < strlen(argv[2]) - 1; i++){

if (argv[2][i] == '1'){

exponent += val;

}

val = val / 2;

}

if (exponent == 0){

mantissa = 0.0;

exponent = -126;

} else {

mantissa = 1.0;

exponent = exponent - bias;

}

// calculate mantissa

float val1 = 0.5;

for(int i = 0; i < 23; i++){

if (argv[3][i] == '1'){

mantissa += val1;

}

val1 = val1 / 2;

}

float res = powf(-1, sign) \* mantissa \* powf(2, exponent);

printf("f - %f\n", res);

// concat string together and then use unsigned int pointer to convert to float

char \*binary;

binary = argv[1];

strcat(binary, argv[2]);

strcat(binary, argv[3]);

unsigned int x = strtoul(binary, NULL, 2);

float \*ptr = (float \*)&x;

printf("f - %f\n", \*ptr);

}

A screenshot of a computer

AI-generated content may be incorrect.