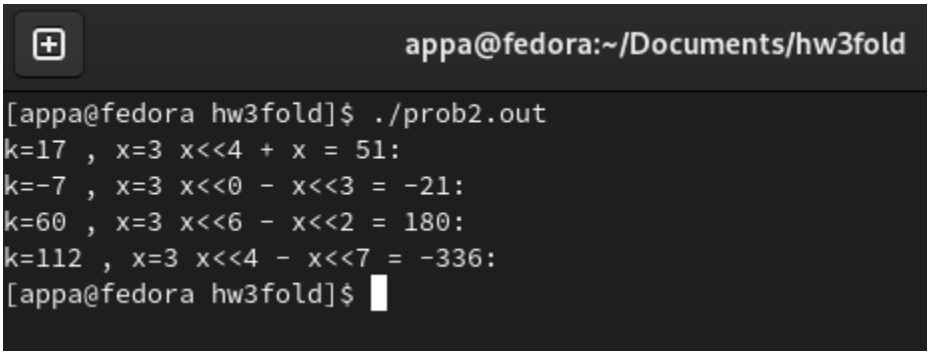
1.

	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

2.

```
#include <stdio.h>
int main (int argc, char **argv) {
    int x=3;
    int exponent;

    exponent = (x << 4) + x;
    printf("k=17 , x=3 x<<4 + x = %d: \n", exponent);

    exponent = (n << 0) - (n << 3);
    printf("k=-7 , x=3 x<<0 - x<<3 = %d: \n", exponent);

    exponent = (x << 6) - (x<<2);
    printf("k=60 , x=3 x<<6 - x<<2 = %d: \n", exponent);

    exponent = (x << 4) - (x<<7);
    printf("k=112 , x=3 x<<4 - x<<7 = %d: \n", exponent);
```

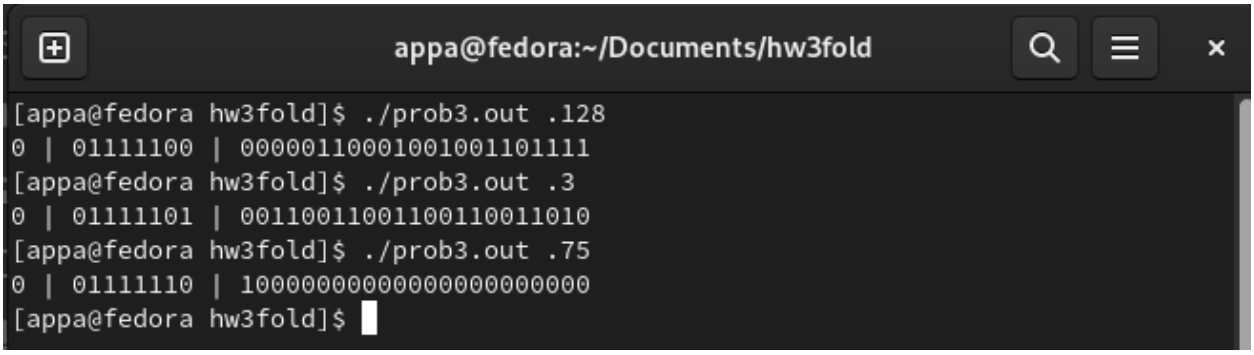
```
    return 0;
}
```



3.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define LEN 32
#define EXP_32 8
#define MAN_32 9

void floatToBinstr(float,char *,int);
void print();

int main(int argc, char **argv) {
    int num=LEN;
    float floatnum;
    char binstr[LEN];

    floatnum = atof(argv[1]);
    floatToBinstr(floatnum,binstr,num);
```

```
	print(binstr, num);


	return 0;
}


void floatToBinstr(float fnum, char *binstr, int num){
  int fling = *(int*)&fnum;
  for (int p = (num-1); p >= 0; p--){
	int len = num-p-1;
	int bit = ((fling >> p) & 1);
	*(binstr+len) = bit + '0';
  }
}


void print(char *binstr, int num) {
	int p=0;
	for (p=0; p < num; p++){
	printf("%c", binstr[p]);
	if(p==0 | p==8)
			printf(" | ");
	}
	printf("\n");
}
```



4.

Value	Binary	Rounded	Action	Rounded
1 1/16	01.00010	01.00	<1/2 down	1
1 3/16	01.00110	01.01	>1/2 up	1 ¼
2 5/16	10.01010	10.01	<1/2 down	2 ¼
2 5/8	10.10100	10.10	<1/2 down	2 ½
3 5/8	11.10100	11.10	<1/2 down	3 ½
3 7/8	11.11100	100.00	>1/2 up	4

5.

Value	Rounded	Exp	Adjusted	Result
256	1.000	8		256
31	10.000	4	1.000/5	32
33	1.000	5		32
35	1.001	5		36
276	1.001	8		288
127	10.000	6	1.000/7	128

6.

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>


int binToDec(int);

double fracDec(char *);
```

```
int main (int argc, char **argv) {
	double s=atoi(argv[1]);
	int exp = binToDec( atoll(argv[2]) );
	double man = fracDec( argv[3] );
	printf("sign=%f, exponent=%d mantissa=%f \n", s, exp, man);
	double floatNum = pow(-1.0, s) *(1 + man) * pow(2, exp-127);
	printf("floating number is %f \n", floatNum);
	return 0;
}

int binToDec(int bin) {
	int dec = 0, p = 0, rem;
	while (bin != 0) {
	rem = bin % 10;
	dec += rem * pow(2, p);
	bin /= 10;
	p++;
	}
	return dec;
}

double fracDec(char* s) {
	double dfactor = 2;
	double dec = 0;
	for (int p = 0; p<23; p++){
	dec += (s[p] - '0') / dfactor;
	dfactor *= 2.0;
	}
	return dec;
```

```
}
```

appa@fedora:~/Documents/hw3fold

```
[appa@fedora hw3fold]$ ./prob6.out 0 01111100 00000000000000000000000
sign=0.000000, exponent=124 mantissa=0.000000
floating number is 0.125000
[appa@fedora hw3fold]$ ./prob6.out 0 01111110 10000000000000000000000
sign=0.000000, exponent=126 mantissa=0.500000
floating number is 0.750000
[appa@fedora hw3fold]$
```