Aufgabe 26a)

#include <stdio.h>

#define INPUT\_TOO\_LONG -1

#define VALID\_INPUT 1

#define NO\_BINARY 2

#define DIM 10

int flush\_buff (void){

int c;

while ((c = getchar ()) != '\n' && c != EOF ){

}

return c != EOF;

}

int read\_binary(int b[]){

int i = 0;

int c = getchar();

while (c != '\n' && i < DIM -1){

b[i++] = c;

c = getchar();

}

if (i == DIM -1 && c != '\n'){

flush\_buff();

return INPUT\_TOO\_LONG;

}

for(i = 0; i < DIM; i++){

if(b[i] != '0' || b[i] != '1'){

flush\_buff();

return NO\_BINARY;

}

}

b[i] = '\0';

return VALID\_INPUT;

}

int main(){

int in[DIM];

printf("Binärcodierung mit %d-Stellen eingeben:\n", DIM);

if (read\_binary(in)== -1){

printf("Eingabe zu lang\n");

}

if (read\_binary(in)== 2){

printf("Eingabe nicht in binaer!\n");

}

if (read\_binary(in)== 1){

printf("Eingabe gespeichert\n");

}

return 0;

}

Aufgabe 26b)

#include <stdio.h>

#include <math.h>

#include <ctype.h>

#define INPUT\_TOO\_LONG -1

#define VALID\_INPUT 1

#define NO\_BINARY 2

#define NAN 3

#define K\_TOO\_HIGH 4

#define DIM 10

double decode\_fk(int festkomma[], int n, int k);

int read\_binary(int b[]);

int flush\_buff (void);

int main(void)

{

int k, in[DIM];

scanf("Anzahl der Nachkommastellen für 10-Bit eingeben\n%d", &k);

if(!isdigit(k)){

printf("Eingabe ist keine Zahl\n");

return NAN;

}

if(k>DIM){

printf("K ist zu hoch gewaehlt!\n");

return K\_TOO\_HIGH;

}

read\_binary(in);

printf("Der decodierte Wert ist: %f\n", decode\_fk(in, DIM, k));

return 0;

}

double decode\_fk(int festkomma[], int n, int k){

double result;

int i;

result = 0;

for(i = 0; i < n; i++){

result \*= 2;

result += festkomma[i];

}

return result / pow(2, k);

}

int read\_binary(int b[]){

int i = 0;

int c = getchar();

while (c != '\n' && i < DIM -1){

b[i++] = c;

c = getchar();

}

if (i == DIM -1 && c != '\n'){

flush\_buff();

return INPUT\_TOO\_LONG;

}

for(i = 0; i < DIM; i++){

if(b[i] != '0' || b[i] != '1'){

flush\_buff();

return NO\_BINARY;

}

}

b[i] = '\0';

return VALID\_INPUT;

}

int flush\_buff (void){

int c;

while ((c = getchar ()) != '\n' && c != EOF ){

}

return c != EOF;

}