Aufgabe 1

(a) **Umwandlung der Dezimalzahlen in duale Festkommazahlen:**
Umwandlung von $(15)_{10}$ in eine Festkommazahl:

$$\begin{aligned} 15 : 2 &= 7 && \text{Rest } 1 \\ 7 : 2 &= 3 && \text{Rest } 1 \\ 3 : 2 &= 1 && \text{Rest } 1 \\ 1 : 2 &= 0 && \text{Rest } 1 \end{aligned}$$

also gilt $(15)_{10} = (1111)_2$.
Insgesamt ergibt sich für die Festkommadarstellung zur Basis 2: $(15)_{10} = (1111)_2$.

Umwandlung von $(6,5)_{10}$ in eine Festkommazahl:
Zuerst die Umwandlung von $(6)_{10}$ in eine Festkommazahl:

$$\begin{aligned} 6 : 2 &= 3 && \text{Rest } 0 \\ 3 : 2 &= 1 && \text{Rest } 1 \\ 1 : 2 &= 0 && \text{Rest } 1 \end{aligned}$$

also gilt $(6)_{10} = (110)_2$.

Nun folgt die Umwandlung von $(0,5)_{10}$ in eine Festkommazahl:

$$0,5 \cdot 2 = 1 \qquad \text{erste Ziffer ist 1, die 1 abziehen ergibt 0}$$

also gilt $(0,5)_{10} = (0,1)_2$.
Insgesamt ergibt sich für die Festkommadarstellung zur Basis 2: $(6,5)_{10} = (110,1)_2$.

Normalisierung:
Es ergibt sich:

$$(15)_{10} = (1111)_2 = (1,111)_2 \cdot 2^3$$

und ebenso

$$(6,5)_{10} = (110,1)_2 = (1,101)_2 \cdot 2^2.$$

Berechnung der Exponenten:
Die Exponenten setzen sich aus dem "echten" Exponenten (hier: 2 und 3) und dem Offset (bei der 32-Bit IEEE 754 Darstellung: 127) zusammen. Es ergeben sich durch Teilen mit Rest:

$$(3 + 127)_{10} = (130)_{10} = (10000010)_2$$
$$(2 + 127)_{10} = (129)_{10} = (10000001)_2$$

Vorzeichen-Bits:
Beide Vorzeichen-Bits sind 0, da es sich sowohl bei $(15)_{10}$ als auch bei $(6,5)_{10}$ um positive Zahlen handelt.

Gleitkommazahl zusammensetzen:
Insgesamt ergibt sich:
Die 32-Bit IEEE 754 Gleitkommadarstellung von $(15)_{10}$ ist:

$$(\underbrace{0}_{\text{Vorzeichen}}\ \underbrace{10000010}_{\text{Exponent}}\underbrace{11100000000000000000000}_{\text{Mantisse}})_{\text{IEEE 754}}$$

Die 32-Bit IEEE 754 Gleitkommadarstellung von $(6,5)_{10}$ ist:

$$(\underbrace{0}_{\text{Vorzeichen}}\ \underbrace{10000001}_{\text{Exponent}}\underbrace{10100000000000000000000}_{\text{Mantisse}})_{\text{IEEE 754}}$$

(b) **Angleichen der Exponenten:**
Der Exponent der Gleitkommadarstellung von $(6,5)_{10}$ ist um 1 kleiner als der Exponent der Gleitkommadarstellung von $(15)_{10}$, da $(10000010)_2 - (10000001)_2 = (1)_2 = (1)_{10}$. Die Mantisse zusammen mit der führenden 1 der Gleitkommadarstellung von $(6,5)_{10}$ wird um eine Stelle nach rechts verschoben, so dass sich $0,11010000000000000000000$ ergibt. (Man beachte: Der Exponent des Ergebnisses ist zur Zeit $(10000010)_2$.)

Addition der Mantissen:
Die Mantisse des Ergebnisses errechnet sich durch

$$\begin{aligned}&(1,11100000000000000000000)_2 + (0,11010000000000000000000)_2\\ &= (10,10110000000000000000000)_2.\end{aligned}$$

Normalisieren/Angleichen des Exponenten:
Das Ergebnis wird normalisiert, indem das Komma um eine Stelle nach links verschoben wird. Der Exponent des Ergebnisses muss deshalb von $(10000010)_2$ auf $(10000011)_2$ erhöht werden.
Die neu berechnete Mantisse ergibt sich somit zu 01011000000000000000000 wobei eine Null am Ende entfällt.

Zusammensetzen des Ergebnisses:
Die 32-Bit IEEE 754 Gleitkommadarstellung des Ergebnisses ist:

$$(\underbrace{0}_{\text{Vorzeichen}}\ \underbrace{10000011}_{\text{Exponent}}\underbrace{01011000000000000000000}_{\text{Mantisse}})_{\text{IEEE 754}}$$

(c) **Multiplikation der Mantissen mit führender 1:**
Es gilt

$$\begin{aligned}&(1,11100000000000000000000)_2 \cdot (1,10100000000000000000000)_2\\ &= (11,00001100000000000000000)_2.\end{aligned}$$

Das Ergebnis muss nun durch Verschieben des Kommas eine Stelle nach links normalisiert werden:

$$(11,00001100000000000000000)_2 = (1,10000110000000000000000)_2 \cdot 2^1.$$

Addition der Exponenten:
Es muss mit den Exponenten ohne Offset gerechnet werden. Diese berechnen sich wie folgt:

$$\begin{aligned}(10000010)_2 - (127)_{10} &= (130)_{10} - (127)_{10} = (3)_{10} = (00000011)_2\\ (10000001)_2 - (127)_{10} &= (129)_{10} - (127)_{10} = (2)_{10} = (00000010)_2\end{aligned}$$

Somit ergibt sich die Summe der Exponenten ohne Offset zu

$$(00000011)_2 + (00000010)_2 = (00000101)_2.$$

Nun wird der Offset wieder zu dem Exponenten addiert:

$$(00000101)_2 + (127)_{10} = (00000101)_2 + (01111111)_2 = (10000100)_2.$$

Da das Ergebnis normalisiert wurde, muss nun noch der Exponent um Eins von $(10000100)_2$ auf $(10000101)_2$ erhöht werden.

Setzen des Vorzeichens:
Da beide Vorzeichenbits der Faktoren 0 sind, hat auch das Ergebnis das Vorzeichenbit 0.

Zusammensetzen des Ergebnisses:
Die 32-Bit IEEE 754 Gleitkommadarstellung des Ergebnisses ist:

$$(\underbrace{0}_{\text{Vorzeichen}}\ \underbrace{10000101}_{\text{Exponent}}\underbrace{10000110000000000000000}_{\text{Mantisse}})_{\text{IEEE 754}}$$

Aufgabe 2

(a) Rechnen Sie von Hand die vierstellige Zahl $(1011)_2$ aus $B = 2$ nach $B' = 10$ mittels Hornerschema um. Schreiben Sie dabei auch explizit die notwendige Klammerdarstellung für die Umrechnung der vierstelligen Zahl $(b_3b_2b_1b_0)$ hin.

Für eine vierstellige Zahl $(b_3b_2b_1b_0)_B$ mit $B \leq B'$ gilt mit Horner-Schema folgende Konvertierung:

$((((b_3) \cdot B + b_2) \cdot B + b_1) \cdot B + b_0)_{B'}$

Für $(1011)_2$ ergibt sich für eine Basenumwandlung von $B = 2$ nach $B' = 10$ also

$((((1) \cdot 2 + 0) \cdot 2 + 1) \cdot 2 + 1)_{10} = 11_{10}$

(b) (i) Wie viele Stellen darf die eingegebene Zahl maximal haben?

99 Stellen, da wir 100 Werte haben, am Ende muss das Terminalsymbol „\0“ stehen.

(ii) Welche Zeilen umschließt das while-Statement und unter welcher Kondition bricht die while-schleife ab?

8 bis 31, solange bis das „\0“-Zeichen gelesen wird. Da in der Schleife am Ende i++ steht, zeigt der Index i beim Test immer auf das nächste Zeichen.

(iii) Was bewirken die Zeilen 12 bis 25?

Konvertierung der beiden ASCII-Zeichen ’0’ oder ’1’ an der momentanen Stelle i im String (von MSB nach LSB gelesen) in Integerzahl 0 bzw. 1, alle anderen Zeichen erzeugen Abbruch.

(iv) Wo findet die eigentliche Berechnung statt?

Zeile 25 berechnet die innere Klammer, während in Zeilen 28-31 die Multiplikation mit B erfolgt, aber nicht für die letzte Stelle b_0.

Kommentierter Code

```c
#include <stdio.h>
int main()
{
  char binaerstring[100];           // Deklariere die Variable fuer die Eingabe
  printf("Gib natuerliche Zahl mit Basis 2 ein: ");
  scanf("%s", binaerstring);        // lies die Eingabe ein
  int i = 0;                        // Initialisiere die Zaehlvariable
  int ergebnis = 0;                 // Initialisiere das Ergebnis
  while(binaerstring[i] != 0)       // Durchlaufe die Schleife so lange, bis die 0 am Ende
                                       des Strings erreicht wird
  {
    char b;                         // Deklariere die Variable fuer die aktuelle Ziffer
    if(binaerstring[i] == '1')      // Wenn das aktuelle Zeichen eine 0 oder 1 ist, setze
                                       b entsprechend
    {                               // b enthaelt nun die k-te Ziffer von vorne: k=0
                                       entspricht dem MSB
      b = 1;
    }
    else if(binaerstring[i] == '0')
    {
      b = 0;
    }
    else                            // Wenn ein anderes Zeichen als eine 1 und eine 0
                                       erkannt wird, Abbruch!
    {
      printf("Ungueltige Eingabe: \'%c\'\n", binaerstring[i]);
      return 1;                     // ..., dann gib den Fehlercode 1 zurueck
    }
    ergebnis = ergebnis + b;        // Klammer-Ausdruck, starte mit b_{N-1}

    i++;                            // erhoehe den Zaehler, Zaehler laeuft vom MSB zum
                                       LSB, genau anders als die Stelligkeit der Ziffer!
    if(binaerstring[i] != 0)        // Wenn die letzte Ziffer noch nicht erreicht ist,
                                       daher nicht bei b_0
    {
      ergebnis = ergebnis * 2;      // dann multipliziere das Zwischenergebnis mit B=2,
                                       aber nicht bei letztem Schritt b_0
    }
  }
  printf("Die Binaerzahl %s entspricht der Dezimalzahl %d.\n", binaerstring, ergebnis);
  return 0;
}
```