```
//Achtung: in der Regel kommentiert man kein Programm so wie hier
//das hier ist nur klausurrelevant, weil es eine schöne Aufgabe zum Kontrollieren ist
#include <stdlib.h>
#include <stdio.h>
//Definition einer struct mit 2 Komponenten wert und nachfolger vom Datentyp int und Zeiger
auf ...
struct Listenelement {
       int wert;
       struct Listenelement *nachfolger;
 };
//mit folgendem Befehl kann ab jetzt statt struct Listenelement
//einfach nur Liste geschrieben werden (was kürzer ist)
typedef struct Listenelement Liste;
//Definition einer Funktion anlegenListe, wo nichts übergeben wird und ein Zeiger,
//der auf Speicher der Grösse Liste zeigen soll, zurückgegeben wird
Liste* anlegenListe(){
   //Anforderung von dynamischem Speicher der Groesse Liste unter Nutzung der Funktion malloc,
   //die in der Bibliothek stdlib definiert ist, Ort des von malloc gefundenen Speichers
   //zuweisen/speichern auf der (Variable) e0
   Liste *e0 = malloc(sizeof(Liste));
   Liste *e1 = malloc(sizeof(Liste));
   Liste *e2 = malloc(sizeof(Liste));
   //auf die Komponente wert der Struktur Liste, wo e0 hinzeigt, wird 0 zugewiesen
   e0->wert = 0;
   //auf die Komponente nachfolger, wo e0...
   e0->nachfolger  = e1;
   e1->wert = 1;
   e1->nachfolger  = e2;
   e2->wert = 2;
   //auf die Komponente nachfolger, wo e2 hinzeigt, wird NULL geschrieben, was ungültige
   //Adresse bedeutet
   e2->nachfolger  = NULL;
   //zurückgeben von e0
   return e0;
}
//Definition einer Funktion ausgebenListe, bei der ein Zeiger vom Typ Liste übergeben wird und
//es wird nix zurückgegeben
void ausgebenDerGanzenListe(Liste *liste){
   liste=liste->nachfolger;//erstes Element ist Kopf, also uninteressant
   //solange der Zeiger liste nicht die Adresse NULL hat
   while(liste != NULL)   {
```

```
      //Aufruf der Ausgabefunktion printf und Ausgabe der Komponente wert des
      //Strukturelements, auf das Variable liste gerade zeigt, auf dem Bildschirm
      printf("Element: %i \n",liste->wert);
      //Variable liste wird der Inhalt der Komponente nachfolger zugewiesen, wo liste eben
      //noch hingezeigt hat
      liste=liste->nachfolger;
   }
   printf("---------------------\n");
}
//Definition einer Funktion ausgebenListe, wo ein Zeiger vom Typ Liste übergeben wird und es
//wird nix zurückgegeben
void ausgebenDerGanzenListeVersionKondring(Liste *liste){
   Liste *ausgewaehltesListenelement=liste;
   liste=liste->nachfolger;//erstes Element ist Kopf, also uninteressant
   //solange der Zeiger liste nicht den Wert NULL hat
   while(ausgewaehltesListenelement != NULL)   {
      printf("Element: %i \n",ausgewaehltesListenelement->wert);
      ausgewaehltesListenelement=ausgewaehltesListenelement->nachfolger;
   }
   printf("---------------------\n");
}
//Einfuegen eines Elements in die Liste vor Stelle position
void einfuegendavorinListe(Liste *liste, int position, int neuerWert){
   int i=1;//null-tes Element ist uninteressant, da Kopf der Liste, deshalb Start beim ersten
Element
   Liste *neuesElement;
   while (liste != NULL){
        //    todo
        if(i == position){
           neuesElement =  malloc(sizeof(Liste));
           neuesElement->wert= neuerWert;
           neuesElement->nachfolger = liste->nachfolger;
           liste->nachfolger = neuesElement;
        }
        liste=liste->nachfolger;
        i=i+1;
   }
}
//Loeschen des Elements an der Stelle position in der Liste
void loescheninListe(Liste *liste, int position){
   int i=1;//erstes Element ist uninteressant, da Kopf der Liste
```

```
   Liste *hilf;
   while(liste != NULL){
     if(i == position){
       hilf=liste->nachfolger;
       liste->nachfolger=liste->nachfolger->nachfolger;
       //Aufruf der Funktion free aus der Bibliothek stdlib.h, um den dynamisch reservierten
       //Speicher unter der Adresse hilf freizugeben, so dass der Speicher nutzbar fuer andere
       //Rechnernutzer ist
       free(hilf);
     }
     liste=liste->nachfolger;
     i=i+1;
   }
}
int main(){
   //Deklaration eines Zeigers zahlenliste vom Typ Liste
   Liste *zahlenliste;

   printf("Erzeugen einer Liste\n");
   //Aufruf der Funktion anlegenliste, Rückgabewert zuweisen (merken) auf Variable zahlenliste
   zahlenliste=anlegenListe();
   ausgebenDerGanzenListe(zahlenliste);

   printf("Einfuegen von 22 vor dem 1.Element der Liste\n");
   einfuegendavorinListe(zahlenliste, 1, 22);
   ausgebenDerGanzenListe(zahlenliste);

   printf("Einfuegen von 33 vor dem 2.Element in der Liste\n");
   einfuegendavorinListe(zahlenliste, 2, 33);
   ausgebenDerGanzenListe(zahlenliste);

   printf("Loeschen des 1.Elements in der Liste\n");
   loescheninListe(zahlenliste,1);
   ausgebenDerGanzenListe(zahlenliste);

   printf("Loeschen des 3.Elements in der Liste\n");
   loescheninListe(zahlenliste,3);
   ausgebenDerGanzenListe(zahlenliste);

   return 0;
```

```
}
```