C-ÜBUNGSAUFGABEN DYNAMISCHE STRUKTUREN 1

Bemerkung:
Die Aufgaben 1) bis 2) beziehen sich auf das in der Präsentation vorgestellte Programm (in dem die “Räume“ in einem während der Laufzeit reservierten Speicherbereich abgelegt werden.
Betrachten Sie den Inhalt des reservierten Speicherbereichs mit dem Debugger.

1)

Kommentieren Sie die Anweisung malloc(...) aus dem Programm aus.
Was geschieht dann während des Programmablaufs ?

2)

Provozieren Sie einen Zugriff auf einen nicht reservierten Speicherbereich.
Was geschieht dann während des Programmablaufs ?

3) Felder variabler Länge verwalten

Es soll ein dynamisches Feld (dynamisches Array) mit Hilfe von Speicherblöcken (mit malloc(...) reservieren) simuliert werden:
Jedes Mal, wenn an das Ende eines Speicherblocks (mit Länge len) ein Element (Integer-Zahl) angefügt werden soll, wird ein Speicherblock der Länge len+1 erzeugt und der Inhalt des alten Speicherblocks (an der Stelle 0 beginnend) in den neuen Speicherblock reinkopiert und an das Ende des neuen Speicherblocks das Element angefügt.
An der Stelle 0 des reservierten Speichers wird immer die Stelle des letzten Elements des dynamischen Feldes gespeichert (**stelleLetztesElement**).
Wenn das dynamische Feld erzeugt wird, hat dieser Zeiger den Wert -1:
stelleLetztesElement = -1
An der Stelle 0 des reservierten Speichers steht also der Wert -1.
Nach jedem An - oder Einfügen wird der Wert von stelleLetztesElement um 1 ehöht, bei jedem Löschen eines Elements aus dem Speicherblock wird der Wert von stelleLetztesElement um 1 verringert.
Hier die konkret zu implementierenden Funktionen:

3.1) Die Methode anuegen(..)

```
int* anfuegen(int *v, int zahl);
```

fügt die Zahl "zahl" an das Ende des "dynamischen" Arrays "v" an.
Wenn eine Zahl angefügt werden soll, muß neuer Speicherblock reserviert werden, der um 1 größer ist als der bisher reservierte Speicherblock.
Dann wird der alte Speicherinhalt (von der Stelle 0 beginnend) in den neuen Speicher kopiert und an die letzte Stelle die Zahl angefügt.
Dann wird der alte Speicher freigegeben.

Wichtig:
An der Stelle 0 des reservierten Speichers wird immer die aktuelle Stelle des letzten Elements des "dynamisches" Arrays eingetragen und verwaltet werden.

Beispiel:
Nach dem Erzeugen (beim ersten Aufruf der Funktion) des dynamischen Feldes soll der Wert 13 eingetragen werden

```
int *feld=NULL;
feld= anfuegen(feld, 13);
```

0	13

Dann soll die Zahl 7 angefügt werden:
Es wird neuer Speicher reserviert. An die Stelle 0 wird die neue letzte Stelle des letzten Elements (also 1, wo die Zahl 7 steht) des neuen dynamischen Feldes eingetragen und die Zahl 13 in den neuen Speicher kopiert. Dann wird noch die 7 an das Ende des dynamischen Feldes eingefügt.

```
feld= anfuegen(feld, 13);
```

1	13	7

3.2) Zur dynamischen Feldverwaltung sollen folgende weietern Methoden gehören:
`void druckeFeld(int *v)`
gibt alle Elemente des dynamischen Feldes auf dem Bildschirm aus.

`void entferneLetztesElement(int *v)`
entfernt das letzte Element des dynamischen Feldes, wobei die Länge des dynamischen Feldes angepaßt (um 1 verringert) wird.

`void entferneAnPosition(int *v , int pos)`
Lösche das Element an der Stelle pos
Beispiel:
dynamisches Feld: 4 5 9 1 3
Lösche Element an der Stelle 2
dynamisches Feld: 4 5 1 3

`public void einfuegenRechts(int *v , int pos, int zahl)`
Für pos muß gelten: -1<=pos <=zeigerLetztesElement
die Zahl zahl wird rechts von pos, also an der Stelle pos+1 eingefügt (nicht überschrieben)
Vorher werden die entsprechenden Zahlen noch nach rechts verschoben.
Beispiel:
dynamisches Feld: 7 5 9 3
Einfügen der Zahl 8 an rechts von pos = 2 ergibt:
dynamisches Feld: 7 5 9 8 3

`public void einsortierenRechts(int *v, int zahl)`
Fügt das Element zahl rechts von der Stelle ein, wo zahl das 1. Mal vom Wert her größer (oder gleich) ist als das dort sich befindliche dynamische Feldelement.
Ansonsten wird es rechts von der Stelle -1 eingefügt.
Beispiel:
dynamisches Feld: 4 5 9 1 3
Rechts einsortieren der Zahl 8 ergibt:
dynamisches Feld: 4 5 8 9 1 3

`public void einsortierenLinks(int *v, int zahl)`
Fügt das Element zahl links von der Stelle ein, wo zahl das 1. Mal vom Wert her größer (oder gleich) ist als das dort sich befindliche dynamische Feldelement.
Ansonsten wird es an das dynamsiche Feldende angefügt.
Beispiel:
dynamisches Feld: 4 5 9 1 3
Links einsortieren der Zahl 2 ergibt:
dynamisches Feld: 4 5 9 2 1 3

```
void entferneZahl(int *v, int zahl)
```
wenn die Zahl zahl das erste Mal im dynamischen Feld vorkommt, wird sie entfernt.
Falls sie nicht vorkommt, passiert nichts.
Beispiel:
dynamisches Feld: 4 1 9 1 3
Entfernen des Zahl 1 ergibt:
dynamisches Feld: 4 9 1 3

```
int sucheZahl(int *v, int zahl)
```
sucht eine Zahl im dynamischen Feld und gibt ihre Position zurück.

```
void sortiereFeld(int *v, int modus)
```
sortiert ein dynamisches Feld auf- oder absteigend (durch Wert des Parameters modus bestimmt).

```
void loescheFeld(int *v)
```
Lösche alle Elemente des dynamischen Feldes

```
void aendereWert(int v[], int wertAlt, int wertNeu)
```
Ersetzt das erste Vorkommen der Zahl wertAlt durch die neue Zahl wertNeu.

```
int loescheDuplikate(int v[])
```
Löscht die Elemente eines dynamischen Feldes, die mehr als 1 Mal vorkommen, so daß jedes Element genau ein Mal im dynamischen Feld vorkommt.
Beispiel:
dynamisches Feld: 7 5 7 9 5 8 9
Duplikate löschen
dynamisches Feld: 7 5 9 8

Bemerkungen:
B1)
In main() sollen alle obigen Funktionen ausführlich (z.B. mit Hilfe von Schleifen) getestet werden.

B2)
Weitere, selbst erfundene Funktionen implementieren.