Klausur Softwareentwicklung 2
SS 2017
1.Termin ITSB

Name: _____ Punkte: 100 Punkte

Personenkennzahl: _____ Zeitvorgabe: 090 Minuten

Gruppe (zutreffende bitte ankreuzen):

BB:	**A**	**B**		
VZ:	**a**	**b**	**c**	**d**
~~**PP:**~~	~~**A**~~			

1. (40) Einlesen-Sortieren-Ausgeben

/*

Ausgehend von folgendem Fileformat, wobei in der ersten Zeile die Anzahl der folgenden Namen-Werte-Paare folgt die Temperatur und der entsprechende Ort:

$ cat zamag_temperaturen.csv

3

9;Salzburg

7;Bregenz

6;Linz

1. Definieren Sie die Datenstruktur und den Datentype (Vorschlag DATA).

Schreiben Sie ein Programm das aus folgenden Funktionen besteht:

2. DATA * readfile(char *filename);

Die Funktion öffnet eine Datei und liest die Werte aus der Datei in ein dynamisches Array mit DATA-Elementen. Die Anzahl der Datensaetzte soll in einer globalen Variablen hinterlegt werden.

3. DATA * sortdata(DATA *p, int element);

Die Funktion sortiert die Daten im übergebenen Array (Zeiger) entweder nach NAME oder nach VALUE aufsteigend. Nach welchem Element sortiert wird entscheidet der Parameter element;

Verwenden Sie die Bibliotheksfunktion qsort:

```
void qsort(void *base, size_t nmemb, size_t size,
        int (*compar)(const void *, const void *));
```

4. void printdata(DATA *p);

Die Funktion gibt die Elemente vom Typ DATA im übergebenen Array (Zeiger) nach stdout aus.

5. int main(int argc, char* argv[]);

Die Funktion übernimmt eine Dateipfad und einen Sortierparameter via Kommandozeilenargument und gibt unter verwendung der Funktionen 1-3 die sortierten Daten wieder auf stdout aus.

ACHTEN SIE BEI ALLEN FUNKTIONEN AUF FEHLERÜBERPRÜFUNGEN!

```
*/
```

2. (20) DE-Queue mittels Liste

/*

Verwenden Sie die folgende Strukturen und realisiere Sie die Funktionen einer dynamischen doppelt verketteten Liste (Double-Ended-Queue). Eine DE-Queue kann sowohl FIFO- als auch LIFO- Verhalten realisieren!

Verwende folgende Strukturen:

```
struct SList {
        struct elem *next;
        struct elem *prev;
        int data;
};
typedef struct SList  SLIST;

struct SList_Header {
  int len;
  SLIST *first, *last;
};
typedef struct SList_Header SLIST_HEADER;
```

Teilen Sie die Definitionen und Deklarationen in Header- und Source-Files. Verwender_innen der DE-Queue sollen folgende Funktionen aufrufen können:

~~**insert_element_at_back** // Fügt hinten eine neues Element hinzu~~
~~**insert_element_at_front** // Fügt vorne eine neues Element hinzu~~
remove_last_element // Gibt das letzte Element zurück und löscht es aus der Liste
~~**remove_first_element** // Gibt das erste Element zurück und löscht es aus der Liste~~
~~**examine_last_element** // Gibt das letzte Element zurück ohne es aus der Liste zu löschen~~
examine_first_element // Gibt das erste Element zurück ohne es aus der Liste zu löschen
get_queue_len // Gibt die Anzahl der Elemente in der Liste zurück

Als Elemente sollen ganze positive Zahlen zwischen 100 und 999 gespeichert werden können. Sorgen Sie dafür dass keine doppelten Werte in der Liste existieren können!

Implementieren Sie die nicht gestrichenen Funktionen!!!

*/

3. (20) Baum-Summe (rekursiv, inorder)

/*

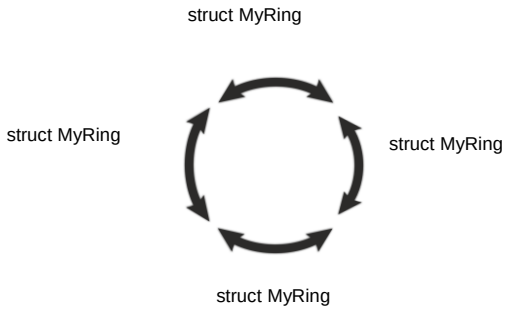
Schreiben Sie die Funktion **int preorder_sum(NODE_PTR)** die die Summe der in den Blättern und Knoten eines binären Baumes gespeicherten int-Werten berechnet indem sie den Baum rekursiv preorder traversiert.

```
typedef struct node{
    struct node *left;
    struct node *right;
    int value;
}NODE;
typedef NODE* NODE_PTR;
```

*/

4. (20) Queue mittels Ring

Verwenden Sie die folgenden Strukturen von Linus Torvalds und realisieren Sie die Funktion zum Einfügen in den Ring und realisieren Sie eine Ausgabefunktion zur Ausgabe aller Elemente im Ring.



Verwende folgende Strukturen:

```
struct list_head{
       struct list_head *next;
       struct list_head *prev;
       }

struct MyRing{
   struct list_head list;
   int  data;
};
```

Realisiere folgende Funktionen:

```
void list_add(struct list_head *new, struct list_head *head)
{









}
void list_print(struct list_head *head)
{









}
```

Folgende main() ist vorgegeben:

```
int main(int argc, char **argv){

   struct MyRing root;            // Leeres Element als Header verwenden

   struct MyRing *tmp;
   struct list_head *pos;



   // Mein Ring initialisieren.
   list_init(&(root.list));

   // Element erzeugen
   tmp = makeNode(7);
   // und in Ring einfuegen
   list_add(                                                    );

   list_print_v1(                                               );

   return 0;
}
```

Viel Erfolg!