S. Kurtz, T. Schwabe,
G. Gonnella, D. Willrodt

Universität Hamburg
Zentrum für Bioinformatik

Programmierung für Naturwissenschaften
Sommersemester 2015
Übungen zur Vorlesung: Ausgabe am 09.04.2015

Punkteverteilung: Aufgabe 2.1: 4 Punkte, Aufgabe 2.2: 6 Punkte

Abgabe bis zum 15.04.2015, 10:00 Uhr.

Aufgabe 2.1 Implementieren Sie eine Funktion `subsetsum`, die das Teilmengen-Summen-Problem löst. Dieses besteht darin, für eine nicht-leere Menge S von positiven ganzen Zahl und eine ganze Zahl k zu entscheiden, ob es eine Teilmenge $S' \subseteq S$ gibt, so dass die Summe aller Zahlen aus S' gleich k ist.

In STiNE finden Sie folgende Dateien, die Ihnen bei der Aufgabe helfen sollen:

- `subsetsum.h`: die Headerdatei zu der von Ihnen zu schreibenden Datei `subsetsum.c`.
- `subsetsum-test.c`: ein Testprogramm, das Ihren Code verwenden soll.
- `Makefile`: ein Makefile um Ihren Code zu compilieren und zu testen.
- `test.out`: die erwartete Ausgabe des Testprogramms.
- `test.sh` und `test.out`: Skript zum Testen
- `int_conv.c` und `int_conv.h`: ein Modul zum Parsen von ganzzahligen Werten aus Strings mit Fehlerbehandlung.

Ihre Funktion sollte wie folgt aussehen und funktionieren (siehe `subsetsum.h`):

```
/* Returns true if the ordered set of numbers <arr> of size <n> contains a
   subset with the sum of its members equal to <k>, returns false if no such
   subset exists. Elements in <mark> will be set to true for all members of
   <arr> that are part of said subset. */
bool subsetsum(unsigned long k,
               bool *mark,
               const unsigned long *arr,
               unsigned long n);
```

Die beiden Arrays `mark` und `arr` haben die gleiche Größe `n` und werden bereits in `subsetsum-test.c` initialisiert.

Wenn Sie alles richtig implementiert haben, dann sollte `make test` fehlerlos durchlaufen.

Aufgabe 2.2 Implementieren Sie in der Programmiersprache C den Datentyp `Queue`. Verwenden Sie die folgenden Konstanten, Typen und Funktionsköpfe aus der Datei `queue.h`.

```
#include <stdbool.h>

typedef int Queueelement;

typedef struct {
  Queueelement *queuespace;     /* the space to store the queue elements */
  unsigned long enqueueindex,   /* points to entry into which element is to be
```

```
                                enqueud */
                dequeueindex,   /* last element of queue */
                queuesize,      /* size of the queue (max no_of_elements)*/
                no_of_elements; /* no of elements between
                                   enqueueindex+1 and dequeindex */
} Queue;

/* The following function delivers an empty queue with a reservoir of
   size elements to be stored in the queue. The reservoir can, if
   necessary, be enlarged. */
Queue *queue_new(unsigned long queuesize);

/* The following function returns true iff the queue is empty. */
bool queue_is_empty(const Queue *q);

/* The following function resizes the queue by doubling the space reservoir  */
void queue_double_size(Queue *q);

/* The following function adds an element elem to the end of the queue. */
void queue_enqueue(Queue *q, Queueelement elem);

/* The following function removes the element elem from the start of the queue.
   */
Queueelement queue_dequeue(Queue *q);

/* print the contents of <*q> on screen */
void queue_print(const Queue *q);

/* The following function frees the space required for the queue. */
void queue_delete(Queue *q);
```

Die Konstanten, Typen und Funktionen haben die folgende Bedeutung:

- Der Typ `Queue` wird zur internen Darstellung der Queue verwendet. Im Array `queuespace` werden alle Elemente der Queue abgelegt. Jedes Element in der Queue ist vom Typ `Queueelement`, der in diesem Beispiel als `int` deklariert ist. Der Speicher für das Array `queuespace` soll je nach Platzbedarf dynamisch mit den Funktionen `malloc` und `realloc` alloziert bzw. erweitert werden (s.u.). Das Attribut `queuesize` soll immer angeben, für wie viele Elemente in der Queue bereits Speicher alloziert wurde. Das Attribut `no_of_elements` gibt die Zahl der aktuell in der Queue enthaltenen Elemente an. Die Attribute `enqueueindex` und `dequeueindex` geben die Positionen im Array `queuespace` an, an denen sich das zuletzt eingefügte Element (Ende der Queue) bzw. das zuerst eingefügte Element (Kopf der Queue) befinden.
- Sollte beim Aufruf der Funktionen `malloc` bzw. `realloc` ein Fehler auftreten, dann wird mit einer Fehlermeldung abgebrochen. Soll ein Element aus einer leeren Queue entnommen werden, dann wird ebenfalls mit einer Fehlermeldung abgebrochen.
- Die Funktion `queue_new` initialisiert eine leere Queue, die maximal `queuesize` viele Elemente aufnehmen kann. `queuesize=0` führt zu einem Fehler.
- Die Funktion `queue_is_empty` liefert `true`, wenn die Queue keine Elemente enthält. Sonst liefert die Funktion `false`.
- Die Funktion `queue_enqueue` hängt ein neues Element mit dem Wert `elem` an die Queue

an. Implementieren Sie die Funktion so, dass beim Einfügen eines Elements kein anderes Element in der Queue verschoben werden muß. Verwenden Sie dazu das Array `queuespace` als zirkuläre Datenstruktur, d.h. Nachfolger der letzten Position in `queuespace` ist die erste Position des Arrays.

Wenn die Queue bereits voll ist, d.h. `noofelements == queuesize`, dann soll die Funktion `queue_double_size` mit Hilfe von `realloc` den Speicher für die Queue verdoppeln. Aufgrund der zirkulären Darstellung des Arrays müssen Sie hierbei ggfs. Elemente innerhalb von `queuespace` verschieben, um ein *Loch* in der Queue zu vermeiden.

- Die Funktion `queue_dequeue` liefert das erste Element der Queue. Dieses steht an Index `dequeueindex` in `queuespace`.
- Die Funktion `queue_print` gibt eine Queue als eine Liste von Elementen, beginnend mit dem zuletzt eingefügten Element auf dem Terminal aus.
- Die Funktion `queue_delete` gibt den für die Queue allokierten Speicherplatz wieder frei.

Testen Sie ihre Queue-Implementierung (die Sie in einer Datei `queue.c` abgelegt haben), mit Hilfe des Programms `queuetest.c`. Das enstehende Programm `queuetest.x` (siehe Makefile `queue_make`) soll beim Aufruf mit dem Parameter 1000, die Ausgabe liefern, die Sie in der Datei `queue1000` finden. Alle die in dieser Aufgabe genannten Dateien (außer `queue.c`) finden Sie auf Stine.

Die Lösungen zu diesen Aufgaben werden am 16.04.2015 besprochen.