Assignment 2

Leveres innen 23 mar innen 23:59 **Poeng** Ingen **Tilgjengelig** etter 26 feb i 8:00

Overview

In this assignment you will implement a map ADT whose interface we have specified for you. A map associates values with keys. The operations to be supported by maps are:

- Creating a new, empty map. (map_create)
- Destroying a may. (map_destroy)
- Mapping a key to an associated value. (map_put)
- Checking whether some key exists. (map_haskey)
- Retrieving the value associated with some key. (map_get)

There should be no bound on the number of elements that may be inserted into the map, short of running out of memory.

Algorithms

The implementation of the map ADT is intended to be based on hashing, as evidenced by the specified map_create() function, which accepts both a comparison function and a hash function. These functions specify how keys in the map are to be compared and hashed, respectively. The main implementation questions are how to deal with collisions (i.e. when two inequal keys have equal hash values) and how to grow the capacity of the map as needed.

Applications

To test your implementation, you will implement a simple utility program called *wordfreqs*. The program should tokenize a set of input files into a sequence of words, and proceed to print all unique words, along with the number of times they occur. The most common word should be listed first, followed by the second most common word, and so on. The input files to process should be specifed on the command-line, like this:

./wordfreqs sometextfile someothertextfile ...

One way to implement this would be to use a map that maps each word to its frequency of occurrence, while keeping a separate list of unique words. When all words have been read, the list of unique words may be sorted in order of descending word frequencies.

Code

Your starting point is the following set of files:

- list.h - Specifies the interface of the list ADT from the first assignment.
- linkedlist.c - An implementation of the list ADT based on doubly-linked lists.
- map.h- Specifies the interface of the map ADT. Do not modify this file.
- common.h - Defines utility functions for your convenience. There are two additions since the previous assignment; a typedef for hash functions, and a function hash_string() which may be used to hash null-terminated strings.
- common.c - Implements the functions defined in common.h.
- wordfreqs.c - An empty stub for the wordfreqs program (complete this).
- Makefile - A Makefile for compiling the code.

As in the first assignment (list ADT), you need to add a new source file that implements your map ADT. Edit the name of this source file into the Makefile, as the value of the MAP_SRC variable.

We've bundled all of the source code in a zip file. The zip file is located in the same folder as this document (p2-pre.zip).

Deadline

This assignment is not mandatory. However, we highly recommend you to complete it.