University of Zurich[UZH]

Department of Informatics
Database Technology Group
Prof. Dr. M. Böhlen

Data Structures and Algorithms
Assignment 6

May 2, 2016

Hash Tables [20 points]

Task 1. [20 points] Implement, in C, a hash table of positive integers with chaining. The hash function to be used is $h(k) = \lfloor m(k \cdot A \bmod 1) \rfloor$ where $A = (\sqrt{5} - 1)/2$ and $k \cdot A \bmod 1$ returns the fractional part of $k \cdot A$.

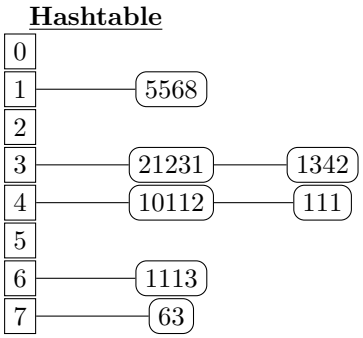
More specifically, your program should include the following:

- *struct element*, that is a struct representing a slot of the hash table.
- *void init(struct element* H[])*, that initializes a hash table `H` of size `m`.
- *int h(int k)*, that gets a key `k` as input and returns the hashed key.
- *void insert(int key, struct element* H[])*, that inserts `key` into hash table `H`.
- *int search(int key, struct element* H[])*, that returns the index of the requested `key` if it is found in the hash table H. Otherwise, it returns -1.
- *void printHash(struct element* H[])*, that prints the table size and all non-empty slots of the hash table `H` accompanied with their index and the keys, as demonstrated in Fig 1.

Test your implementation using a hash table of size 8.

- Add the values 111, 10112, 1113, 5568, 63, 1342 and 21231.
- Print the resulting hash table.
- Search for values 1, 10112, 1113, 5568, 337 and print the results.

Note: Mathematical functions that are available via library `math.h` can be used and do not need to be redefined.



Output printHash

```
Table size: 8
i: 1       key: 5568
i: 3       key: 21231 -> 1342
i: 4       key: 10112 -> 111
i: 6       key: 1113
i: 7       key: 63
```

Figure 1: printHash output

Dynamic Programming [30 points]

Maximum Wine Profit

Task 2. [20 points] Assume you have a collection of N wines placed next to each other on a shelf. Integers from 1 to N are attributed to the wine bottles from left to right and the price of the i^{th} wine is notated as p_i (prices of different wines can be different).

Because the wines get better every year, supposing today is the year 1, on year y the price of the i^{th} wine will be y*p_i.

You want to sell all the wines you have, but there are the following two constraints:

1. You can sell exactly one wine per year, starting this year.
2. Each year you are allowed to sell only either the leftmost or the rightmost wine on the shelf and you are not allowed to reorder the wines on the shelf (i.e. they must stay in the same order as they are in the beginning).

You need to determine the maximum profit you can get, if you sell the wines in optimal order.

Example: If the prices of the wines are (in the order as they are placed on the shelf, from left to right): p_1=1, p_2=4, p_3=2, p_4=3. The optimal solution would be to sell the wines in the order p_1, p_4, p_3, p_2 for a total profit $1*1+3*2+2*3+4*4=29$.

Solve the problem using (a) recursion, (b) memoization and (c) dynamic programming. Create a C program including the following:

- The constant *MAX_N* equal to the maximum number of elements available in any of the wine profit problems you will test.
- A global matrix *m[MAX_N][MAX_N]* used in order to store the intermediate results when memoization or dynamic programming is used.

- The function *int wineprofitRecursive(int price[], int n, int begin, int end)* that computes the maximum profit recursively.
- The function *int wineprofitMemoized(int price[], int n, int begin, int end)* that computes the maximum profit using memoization.
- The function *int wineprofitDynamic(int price[], int n)* that computes the maximum profit using dynamic programming.

Using the following test sets, print out the maximum profit and the memoized matrix `m` at the end:

Test	**N**	**prices**
A	4	1,4,2,3
B	5	2,3,5,1,4

Task 3. **[10 points]** This task is about comparing the efficiency of 3 different algorithms solving the problem of Task 2. You are given 3 datasets "small.txt", "medium.txt", and "large.txt" representing three instances of the problem "Maximum Wine Profit". Each line of the dataset contains an integer that represents a wine price The first line represents the price of the leftmost bottle of wine on the shelf and the last line the rightmost one. "small.txt" and "medium.txt" contain only 30 and 35 lines respectively, while "large.txt" contains 10000 lines. Run your implementations of `wineprofitRecursive`, `wineprofitMemoized`, and `wineprofitDynamic` on these datasets and report the elapsed time as following:

	small (30)	medium (35)	large (10000)
Recursive			
Memoirzed			
Dynamic			

Note: The recursive algorithm needs more than an hour to solve problem instances with more than 40 prices. Thus, avoid applying the recursive solution for "large.txt".

Submission

For this exercise, you need to submit a zipped folder *a<exercise number>_<family name>_<matriculation number>.zip* where `family name` and `matriculation number` correspond to your personal data. This folder should include the C-files you created for each of the tasks. Each C-file should be named as *task<task number>.c* and it should also include you personal data in the form of a comment on the top.

Deadline: **Sunday, May 15th at 23:59**.