University of Zurich[UZH]

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

Informatik II
Assignment 4

Apr 3, 2017

Linked Lists [20 points]

Task 1. [10 points] Write a C program that implements singly-linked lists of integers defined as follows:

```
struct list {
  struct node* head;
};
```

```
struct node {
  int val;
  struct node* next;
};
```

Your program must contain the following functions:

a) *struct list* init()* that initializes a list.

b) *void append(struct list *listA, int val)* that appends a new node containing integer `val` at the end of the list.

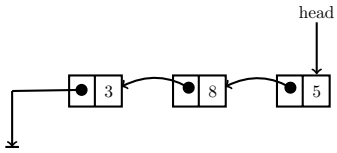
c) *void print(struct list *listA)* that prints the values of the elements of the list to the console enclosed in brackets, e.g. `[ 3 5 7 2 ]`.

d) *void delete(struct list *listA, int i)* that removes the node in the *i-th* position in the list. The index of the first element is 0.

After you have implemented and tested these functions with lists of your choice, include the following sequence in your *main()* method:

1. Insert 9,4,5,3,1,2,0 to the list, in the given order.
2. Print the list to the console.
3. Remove the nodes in positions 6, 3 and 0 from the list, one after another.
4. Print the list to the console.

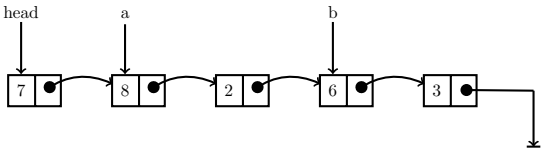
Task 2. [**6 points**] A singly-linked list of pointers can be used to represent decimal numbers in place-value notation. In such a scheme, the first node of the list represents the least significant digit, the second node the second-least significant digit and so on. For example, the list below represents number 385.



Write a C program that uses a function *struct list* addTwoLists (struct list* numA, struct list* numB)* to add the two decimal numbers represented using the lists *numA* and *numB* and returns a pointer to a new list that contains the result of the addition.

Task 3. [**4 points**] Assume a function `swap(struct list* listA, struct node* a, struct node* b)` that receives a singly-linked list, that you implemented in Task 1, and two pointers (`a` and `b`) to its nodes and swaps the nodes `a` and `b` by **modifying the appropriate pointers instead of modifying their node values**.

1. Redraw the following list after function `swap` has been applied on the given pointers `a` and `b`.



2. Assume that you need to implement function `swap`. Can we omit the first argument (`struct list* listA`) of function `swap`? Would it be still possible to swap nodes `a` and `b` by modifying the appropriate pointers? Justify your answer.

Abstract Data Types (ADT) [20 points]

Task 4. [**10 points**] Implement a FIFO `queue` of positive integers that is able to store up to `16` elements and is defined as follows:

```
struct queue {
  int arr[16];
  int head;
  int tail;
};
```

Along with the above datatype, create the following functions:

- *struct queue* init()* that initializes a queue.
- *int isempty(struct queue* q)* which checks if queue `q` is empty or not.
- *void enqueue(struct queue* q, int val)* which inserts integer `val` into `q`.
- *int dequeue(struct queue* q)* which removes the proper element from `q` and returns the value of the element removed. In case of an error, -1 should be returned.
- *void print(struct queue* q)*, which prints the integers in queue `q`.

Test your implementation by performing the following operations:

- Create queue `q`.
- Enqueue 4, 7, 1, 2.
- Print `q`.
- Dequeue two elements and print their values.
- Print `q`.

Task 5. [10 points] The LIFO behaviour of a stack can be simulated using two FIFO queues. The elements of the stack are moved from one queue to another to properly perform the *push* and *pop* operations. After the completion of a *push* and *pop* operation, all elements are stored in only one of the two queues.

a. Assume that a stack of size 16 is implemented using two queues of size 16 each. Initially the queues, and thus the stack, are empty. Make a table to illustrate the sequence of queue operations needed to simulate the following sequence of stack operations: `push(20); push(5); pop(); push(15);` Draw the state of the queues after each stack operation.

b. Implement a stack datatype that can store up to 16 elements, using two queues that can store up to 16 elements each. Along with the datatype provide the following functions:

   - *struct stack* initStack()*, that initializes a queue.
   - *void push(struct stack* s, int val)*, that inserts integer `val` onto the stack.
   - *void swap(struct stack *s)*, that swaps the pointers of the two queues.
   - *int pop(struct stack* s)*, that removes an element from the stack.
   - *void printStack(struct stack* s)*, that prints the stack.

   Test your stack with the following sequence of operations:

   - Push 8, 9, 12, 3, 5 onto the stack.
   - Print the stack.
   - Pop three elements and print their values.
   - Print the stack.
   - Push 11, 4, 21, 6, 2, 5, 13, 14.
   - Print the stack.

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:

a) the C-files you created for each of the tasks. Each C-file should be named as *task<task number>.c*

b) a pdf named *a<exercise number>.pdf* with the solutions for Tasks 3 and 5a.

Make sure that both in the C-files as well as in the pdf file you submit, your personal data is included (in the form of comments or a note).

Notes

- You are allowed to create as many additional functions as you need in all exercises and you can also reuse material from previous exercises if needed.

Deadline: **Sunday, April 23rd at 23:59**.