VG101 — Introduction to Computer and Programming

Assignment 6

Manuel — UM-JI (Fall 2017)

- MATLAB: write each exercise in a different file
- C/C++: use the provided assignment template
- Include simple comments in the code
- If applicable, split the code over several functions
- Extensively test your code and impove it
- Write a single README file per assignment
- Zip all the files and upload the archive on Canvas

Group Exercise

The goal of this exercise is to get a better understanding of pointers, while helping each others. For a better result please apply the following suggestions.

- Start thinking of the problem as early as possible;
- Relate linked lists to arrays and think of the differences;
- Think of the difference between inserting/deleting the first element and the last element;
- For a total of nine functions, each student is expected to write three. The workload distribution should be detailed in the README file;

Remark: linked lists are a very common data structure, and many implementations are available online. Do not reuse any code from others; Honor Code will be strictly applied.

Ex. 7 — *Linked lists*

1. Online research questions.
   a) Explain what a linked list is?
   b) List some applications of linked lists.
   c) Search what kinds of linked list exist.
2. Programming questions.
   a) The code provided below has been "compacted" in order to fit over a single page. Reorganise it writing no more than one instruction per line while respecting indentation.
   b) Based on the header file below complete the implementation of the linked list.

Linked list header file (ex. 6)

```c
#ifndef LIST_H
#define LIST_H
typedef struct node{ char ch; struct node *next; } node_t;
typedef enum{false, true} bool;
node_t *Initialize(char ch);
void PrintList(node_t *head);
void FreeList(node_t **head);
bool IsEmptyList(node_t *head); // Return true if the list is empty, false otherwise
void InsertFirstList(node_t **head,char insert_char); // Prepend a node
void InsertLastList(node_t **head,char insert_char); // Append a node
void DeleteFirstList(node_t **head); // Delete the first element in the list
void DeleteLastList(node_t **head); // Delete the last element in the list
```

```c
int SizeList(node_t *head); // Return the size of the list
int SearchList(node_t **head, char target); // Count how many times target appears
void SplitList(node_t **head, node_t **tail, int pos); // Split into [0;pos-1] and [pos,end]
void MergeList(node_t **head1, node_t **head2); // Merge two lists
#endif
```

Linked list implementation (ex. 6)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "list.h"
int ex6() {
    node_t *a=Initialize('1'); node_t *b=NULL; PrintList(a);
    InsertFirstList(&a, 'V'); InsertFirstList(&a, 'M');
    PrintList(a); InsertLastList(&a, 'C'); PrintList(a);
    SplitList(&a, &b, 2); PrintList(a); PrintList(b);
    DeleteFirstList(&a); PrintList(a); InsertLastList(&a, 'G');
    DeleteLastList(&b); PrintList(b); InsertLastList(&b,'0');
    PrintList(b); InsertLastList(&b, '1'); PrintList(b);
    MergeList(&a,&b); PrintList(a);
    char target='G';
    printf("Count '%c': %d\n",target, SearchList(&a,target));
    target='1';
    printf("Count '%c': %d\n",target, SearchList(&a,target));
    FreeList(&a);
    return 0;
}
node_t *Initialize(char ch) {
    node_t *head;
    head=(node_t*)calloc(1,sizeof(node_t));
    if(head==NULL){ fprintf(stderr,"Failed to assign memory!\n"); exit(-1); }
    head->next=NULL; head->ch=ch;
    return head;
}
void PrintList(node_t *head) {
    node_t *temp=head;
    printf("***Print Linked List***\n");
    while(temp!=NULL) { printf("%c ",temp->ch); temp=temp->next; }
    printf("\n****Print Finished****\n\n");
}
void FreeList(node_t **head) {
    node_t *tmp=NULL; node_t *pHead=*head;
    while(pHead->next!=NULL) { tmp=pHead; pHead=pHead->next; free(tmp); }
    free(pHead);
}
```

Expected output (ex. 6)

```
$  ./h6 -ex6
**Print Linked List***
1
***Print Finished****

***Print Linked List***
M V 1
***Print Finished****

***Print Linked List***
M V 1 C
***Print Finished****

***Print Linked List***
M V
***Print Finished****

***Print Linked List***
1 C
***Print Finished****

***Print Linked List***
V
***Print Finished****

***Print Linked List***
1
***Print Finished****

***Print Linked List***
1 0
***Print Finished****

***Print Linked List***
1 0 1
***Print Finished****

***Print Linked List***
V G 1 0 1
***Print Finished****

Count 'G': 1
Count '1':  2
```