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

Ex. 1 — *Structure, pointers, and functions*

A point P in the complex plan can be defined using either Cartesian or Polar coordinates.

1. Write two structures to represent a point in the complex plan. In the README file explain and argue on your choices, including data types.
2. Write two functions to convert from Cartesian to Polar and from Polar to Cartesian. The functions should use pointers such that more than one set of coordinates can be converted at a time. That is the pointer could contain more than one complex number such that the function would returns the conversion of all the complex numbers at once.
3. In the main function define the following complex numbers and convert them between Cartesian and Polar coordinates: $3+\frac{4}{5}i$, $\log(4)i$, $45.245+0.235i$, $3e^{\frac{i\pi}{17}}$, $4(\cos\frac{\pi}{9}+i\sin\frac{\pi}{9})$, $e^{\frac{i\pi}{12}}$.

Specifications.

- Organise the complex numbers into two arrays: Cartesian and Polar
- Prints the numbers by pair, one pair per line, the original number first and the converted one second

Ex. 2 — *Pointers, loops, and conditional statements*

Write a C program that reads through an array of integers using pointers, and scan through it to find all the values larger than a randomly generated number r. The element r must be smaller than the max of the array.

Specifications.

- Start by reading the number of integers n
- Then read the n space separated integers
- On a new line display the output as space separated elements

Ex. 3 — *Strings and file I/O*

Write a program to find the number of times a given string occurs in a sentence. The program should read the sentence from a file called `sentence.txt` and the word from a file called `word.txt`. The output should be printed in a file `count.txt`.

Specifications.

- The binary, input, and output files are expected to be in the same directory
- Do not use absolute paths
- Do not prompt the user

Ex. 4 — *File I/O, arrays, and loops*

Read two matrices A and B from a text file called `matrices.txt`. Compute $A + B$, $A \cdot B$ and $A^T \cdot B^T$. Output the result into a text file named `result.txt`. Each row of a matrix is represented as a list of integers separated by a space. When the end of a row is reached a new one starts on the next line. Two matrices a separated by a blank line.

Specifications.

- The binary, input, and output files are expected to be in the same directory
- Do not use absolute paths
- Do not prompt the user
- Output the resulting matrices in the order "addition, multiplication, transpose"

Partial sample output (ex. 4)

```
$ ./h6 -ex4
1 2 3
2 3 4
5 6 7

9 8 7
4 3 2
5 7 2
```

Ex. 5 — *Basic object oriented programming in C*

A mathematical set is a collection of distinct objects, such as $\{1, 3, 9\}$, $\{r, g, b\}$, and $\{5, 11, 11\} = \{5, 11\}$. Based on the multi-set header file below write in a corresponding `set.c` file the necessary functions to handle a set (creation, deletion as well as adding and removing elements). Assume a set can contain elements of type either `char`, `int` or `double`.

Hint: to resize a memory block use the function `realloc`.

Universal set header file (ex. 5)

```c
#define INITSETSIZE 64 // Initial memory allocated for the set
#define CHAR 1
#define INT sizeof(int)
#define DOUBLE sizeof(double)
/* elem: list of elments; card: cardinal of teh set; type: data type (CHAR INT or DOUBLE) */
typedef struct universalSet { void *elem; int card; int type; } uset;
/* Initialize an empty set of given type and allocate the initial memory: INITSETSIZE*type */
void newSet(uset *set, int type);
void deletSet(uset *set); // Free the memory allocated by newSet
/* add elem to the set: check whether it is already in the set;
   resize memory if card = allocated memory; new memory = previous+64
   e.g. before: mem=128, card=128, after: men=192, card=129 */
void addElem(void *elem, uset *set);
/* remove elem from the set; do nothing if the set does not contain this elem;
   resize memory if "too much memory" is used; new = previous-64
  e.g. before: mem=192, card=129, after: card=128, mem=128 */
void remElem(void *elem, uset *set);
```

Ex. 6 — *Coding quality*

Review the previous assignment comparing your code to the samples provided by the TAs. Precisely and clearly list all the aspects where you need to improve. The list can be in the form of bullet points and must be written in a separate text format file.

Note: common needed improvements relate to the README file, the quality of the algorithm, the presentation of the code (spacing, indentation etc.), use and naming of the variables, etc.

Details each aspect that needs improvements and provide clear guidelines for leading to higher coding quality.

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)

```
#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
```

Groups

Cao Yifan, 曹依凡 (517370910097)
Pan Zhiyi, 潘芝亦 (517370910040)
Xiang Yi, 项翊 (517370910204)
Shen Yiming, 沈亦明 (517370910254)

Wang Yichao, 王逸超 (517370910011)
Zhao Shaochong, 赵绍充 (517370910125)
Gao Yang, 高扬 (516141910036)
Chi Pengnan, 迟朋南 (517370910245)

Zhou Zikun, 周子琨 (517370910209)
Zhou Yihua, 周逸华 (517370910154)
Hou Yichun, 侯一纯 (517370910128)
Yu Yang, 俞洋 (517370910092)

Qian Xinyu, 钱心宇 (517370910056)
Jin Junhui, 金峻辉 (517021910702)
Merle Braatz (715370990065)
Wang Jiaqi, 王佳祺 (517370910228)

Bai Yuyang, 白豫阳 (517370910077)
Wang Xiheng, 王希恒 (517021910095)
Jinyou Kim (517370990020)
Liang Zihe, 梁子赫 (517370910051)

Ge Jintian, 葛劲天 (517021911142)
Zhang Junxiang, 张隽翔 (517370910064)
Jeehun Chung (517370990036)
Gu Yile, 顾以勒 (517370910109)

Shao Xuesi, 邵学思 (517370910172)
Jiang Xinmeng, 蒋昕萌 (517370910239)
Liang Xinyu, 梁新宇 (517021911149)
Claudia Jorda Terrado (517370990031)

Shinedul Purevdorj (517370990007)
Doo Ho Ro (517370990026)
Tang Yuchen, 唐瑜辰 (517370910117)
Tong Xinyu 童新雨 (517370910101)

Feng Ruiquan, 冯睿泉 (517370910048)
Zhu Wenxuan 朱文轩 (517370910263)
Zhou Shu, 周澍 (516204910015)
Tao Chenyun, 陶晨韵 (517370910072)

Zhang Shengan, 张圣安 (517370910153)
Ji Duohui, 姬铎辉 (5143709252)
Ali Poursani (517370990033)
Xia Binyu, 夏彬禹 (517370910012)

Zhu Zhuoer, 朱卓尔 (517370910066)
Chen Mengxuan, 陈梦璇 (517370910155)
Fan Zekai, 范泽楷 (517021911109)
Fang Jingzhe, 方靖哲 (517370910018)

Zhao Yuntian, 赵云天 (517370910005)
Wu Xincheng, 吴心澄 (517370910089)
Funglui Koo 峰 (517370910247)
Ye Shunyi, 叶顺义 (517370910091)

Zhou Wenhan, 周问寒 (517370910126)
Shen Mengyuan, 沈梦圆 (517021910139)
Philipp Dolle (715370990057)
Tang Xinyi 唐心怡 (517370910021)

Ding Rui, 丁锐 (517021910642)
Li Dinan, 李迪南 (517370910168)
Yin Xinlong 尹鑫龙 (517370910152)
Ye Xinyi, 叶歆怡 (517370910102)

Ma Jiaxiang, 马嘉祥 (517021911117)
Zhang Shuo, 张硕 (517370910181)
Han Xu, 韩旭 (517021911111)
Song Yanbo, 宋彦伯 (517370910058)

Dongbin Park (517370990016)
Yuan Yichao, 袁意超 (517370910233)
Fei Jiani, 费佳妮 (517370910037)
Huang Ziyuan, 黄梓渊 (517370910250)

Miao Zehao, 缪泽浩 (517370910055)
Wu Shiyuan, 吴诗媛 (517370910073)
Jiang Wenhao, 姜文浩 (517370910081)
Sun Woo Kim (517370990015)

Cen Sheng, 岑盛 (517370910045)
Ma Ziqiao, 马子乔 (517370910114)
Zhang Dingkun, 张定坤 (517370910261)
Chen Tianyu, 陈天羽 (517370910137)

Nanshi Ko (517370990032)
Joshua Wayne Chang (517370990011)
Hao Zhiyu, 郝知雨 (517370910098)
Jiwon Yun (517370990006)

Taeguek Ha (517370990010)
Cai Xianjiao, 蔡先觉 (517370910193)
Yang Yiwei, 杨亦纬 (517370910063)
Ji Guanying, 吉莞颖 (517370910185)

He Zhengfei, 何正非 (517021911112)
Ran Shuangxuan, 冉双璇 (517021910003)
Ma Yiwen, 马逸文 (517370910171)
Huang Yihao, 黄奕豪 (517370910026)

Yang Yuao, 杨雨澳 (517021910098)
Xu Yihang, 徐逸航 (517370910120)
Lu Xingyu, 陆星宇 (517021911116)
Xu Yiyang, 徐绎洋 (516370910071)

Chen Qingyi, 陈清逸 (517370910079)
Wang Jing, 王婧 (517370910133)
Cao Kaiyan, 曹升岩 (517370910244)
He Xiying, 何禧 (517370910001)

Zheng Zhirui, 郑智睿 (517370910044)
Tao Rong, 陶榕 (517370910148)
Fu Tianchi, 傅天驰 (517370910007)
Feng Kai, 冯凯 (517370910047)

Elliot Sujong Lee (517370990034)
Zhang Xiuqi, 张修齐 (517370910208)
Khawaja Zohaib Shariq (517370990035)
Tu Yuyue, 屠余岳 (517370910173)

Li Haochen, 李浩辰 (517370910009)
Lei Jiaqi, 雷嘉琦 (517370910038)
Wang Ruobing, 王若冰 (517021911106)
Lu Jiachen, 卢嘉晨 (517370910030)

Deng Yanhao, 邓雁灏 (517021910661)
Yao Yuan, 姚渊 (517021911184)
Tu Xun, 涂勋 (517370910149)

Awan Osama Malik (516370990003)
Pan Qiying, 潘启颖 (517370910039)
Qian Cheng, 钱程 (517370910225)

Gracia Stefani (517370990022)
Bao Yufan, 包 (517370910067)
Yang Xiao, 杨逍 (517370910062)