ENGR151 — Accelerated Introduction to Computer and Programming

Homework 6

Manuel — UM-JI (Fall 2021)

- 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 homework
- Carefully follow the git submission process

*Exercises preceded by a * are mandatory.*

JOJ Online Judge

Exercises 2 to 7 can be tested on JOJ Online Judge.

Important reminders regarding the Online Judge (OJ):

- For each exercise save all the files, without any folder structure, into a `.tar` archive;
- Strictly stick to the input and output formats provided in the specifications;
- The OJ only checks the correctness of the code not its quality;
- For feedbacks on the quality, submit the code as part of the assignment and include the OJ score as well as the failed cases in the README file;

Ex. 1 — *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. 2 —** *Structure, pointers, and functions*

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

1. Write two structures to represent a point in the complex plane. 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
- Input: none
- Output: one pair per line, the original number first and the converted one second, display five decimals (e.g. 3.00000+0.80000i 3.10483e0.26060i)

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
- JOJ extra rules:
  - Add `#include "assignment.h"`
  - Do not include the `main()` function
  - The function prototype should be `void ex3()`

* **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"
- JOJ extra‘ rules:
  - Add `#include "assignment.h"`
  - Do not include the `main()` function
  - The function prototype should be `void ex4()`

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 —** *Simplified dynamic arrays and 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 universal set header file below write in a corresponding `ex5.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`.

Specifications.

- JOJ extra rules:
  - Add `#include "universal_set.h"`
  - Do not include the `main()` function

Universal set header file (ex. 5)

```c
#ifndef USET_H
#define USET_H
#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 the 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);
#endif
```

Ex. 6 — *Conditional statements, loops, pointers, ASCII code*

Write a C program where the user inputs a string and is then offered a menu to replace or delete a character of his choice. The program should then display the initial string with the requested changes applied. When deleting a character use the functions `memmove` or `memcpy`.

Specifications.

- Input: three lines, a string on the first, a number on the second, and a character on the third
- Output: one line with the result

Sample output (ex. 6)

```
Input a string: good morning
* Choose 1 to replace a character or 2 to delete a character: 1
  Replace character: g
  with: d
New string: dood mornind
```

Pair programming

The goal of the following exercise is to practice programming as *a pair*. For a better group work experience the following scenario is recommended.

- Sit in a comfortable environment and work together as a team;
- A student plays the "Driver" and the other one the "Navigator";
- The *driver*'s work is to type on the keyboard while the *navigator* provides suggestions;
- Both the *driver* and the *navigator* should pay attention to common typos and errors;
- Roles can be exchanged after a while;
- Both students are expected to think of the whole problem;

* **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.

Specifications.

- JOJ extra rules:
  - Ensure you have `#include "linked_list.h"`
  - Do not include the `main()` function

Linked list header file (ex. 7)

```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
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. 7)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "linked_list.h"
int ex7() {
    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. 7)

```
$ ./h6 -ex7
**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
```