Exercises and assignments, C-programming

Work package nr 3/ Extended C-programming (Max 10 p)

For this week tasks we will develop some, a little bit more advanced general C-programs.

Exerc_3_ 1 (Filename exerc_3_1.c) (2p)

Implement a test program for a robot. The program asks for the robot's starting position (x, y coordinates, range 0-99) and then **for a string of characters 'm' and 't'**, where **m** stands for move one step in current direction and **t** for turn of direction as below.

move:	means that the robot takes one step in the current direction.
turn:	means that the robot turns 90 degrees clockwise. Start direction is always north.

The program performs the instructions of the string one by one. When all instructions are executed robot stops and the program prints out the new robot position for the robot. The program then asks for new starting position, etc.

Implement the functions move() and turn() as two void functions and use a pointer parameters as arguments so that the function can update the robot position which is a variable in the main function (calling function).
Use enum and a record of type ROBOT as below for the robots position and direction.

```
 enum DIRECTION {N,O,S,W};

typedef struct {
    int xpos;
    int ypos;
    enum DIRECTION dir;
} ROBOT;
```

Exerc_3_ 2 (Filename exerc_3_2.c) (2p)

All sub tasks in this exercise (Searching and sorting) should be implemented and tested in the same program.

a) Write a function that given an integer **n**, an array of integers and the size of the array determines if **n** is in the array. If so the function should return the index for the first position of the number (in case of several) otherwise returns -1.
For testing the function, write a main program that tests the function with help of an array initiated in the main program as below and with a function declaration:

```
int search_number( int number, int tab[], int size);
int test [] = { 1,2,34,5,67,3,23,12,13,10};
```

b) There are a lot of ways to sort an array. For example, bubble sort which not is the fastest, but it is easy to understand and implement. Write a sorting routine that use the following algorithm to sort an array of integers.

• Find the minimum value in the list.
• Swap the minimum with the first in list.
• Repeat this but exclude the previous minimum on top of the list and search only in the rest of the list.

Implement the sorting function using the function declaration:

void sort (int number, int tab []);

Test the function by use of a main program and an initiated array as above. For checking purpose print out the sorted array.

Exerc_3_ 3 (Filename exerc_3_3.c) (2p+1p)

a) Write a function that creates a linked list with a NUMBER of records of type REGTYPE (see below). The value of the variable data is given a random number between 0 and 100.

Function declaration : **REGTYPE * random_list (void);**
Complete the program with a main program that tests the function (a first draft below).

b) Extend the program with a function with the function declaration:
REGTYP * add_first (REGTYPE * temp, int data);
That adds a new record first in the list and assign the field *numbers* the value of *data*.
The function must return a pointer to the new first entry in the list. Extend main so that this function is tested.

```
/*********************************************
    DIT1165 Program     file exerc_3_3.c               **
    2018-01-04                                         **
*********************************************/
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

//#### Konstanter #####
#define MAX 5

// ##### Typedefs         ####
typedef struct q{
      int number;
      struct q *next;
      struct q *prev;
} REGTYPE;

// ##### Funcion declarations   #####

REGTYPE* random_list(void);
REGTYPE* add_first(REGTYP* temp, int data);




//###### Main program #######
int main(int argc, char *argv[])
{
   int nr=0;
   REGTYPE *akt_post , *head=NULL;

   srand( time(0));  //   Random seed
   head=random_list();
   akt_post=head;
   while( akt_post!=NULL){
         printf("\n Post nr %d : %d" , nr++, akt_post->number);
         akt_post=akt_post->next;
   }

     ………
     ………

 // --- Free of allocated memory  ---

  while((akt_post=head)!=NULL){
    head=akt_post->next;
    free(akt_post);
  }

  //-----------------
  return 0;
}
// ====     End of main   ======================================
REGTYPE* random_list(void ){
   int nr,i=0;
   REGTYP *top, *old, *item;
   ……….


   return(top);
}
```

```
 //===========================================================
REGTYPE* add_first(REGTYPE* temp, int data){
// Adds a record first i list and set the field tal to data


}
```

Exerc_3_ 4 (Filename exerc_3_4.c) (3p)

File managements of a person register

You should write a program for manage a database of people. The database should be stored to the hard disc as a binary file. The function of the program is easiest to understand by reading the description and program skeleton below.

From the **main program** you should be able to choose between these options:

1 Create a new and delete the old file.
2 Add a new person to the file.
3 Search for a person in the file .
4 Print out all in the file.
5 Exit the program.

After entered the choice the program executes the task and returns to the menu for new choices.

1. Create a new and delete the old file.
   Program creates a new file with the specified filename (fixed) and writes a first dummy record to the file and then close it.

2. Add a new person to the file.
   First gives an opportunity to put in one new person to a temp record and then add this record in the end of the file.

3. Search for a person in the file.
   Gives you an opportunity to search for all persons with either a specified first name or family name (by choice).The program prints out all person with that name.

4. Print out all in file.
   Prints out the whole list

5. Exit the program.
  Just exits the program.

```
/ * ====================================
File name: exerc_3_4.c (or cpp)
Date: 2019-mm-dd
Group Number:xxxx
Members that contributed:
…..
Demonstration code: [<Ass code 1-3> <abc>]       Important !
====================================== * /

#include <stdlib.h>
#include <stdio.h>

// -----Typedefs -------
typedef struct {
       char firstname[20];
       char famnamne[20];
       char pers_number[13]; // yyyymmddnnnc
}PERSON;

// Function declaration (to be extend)
PERSON input_record( void);          // Reads in a person record.
void write_new_file( PERSON *inrecord); //Creats a file and write a first record
void printfile(void); // print out all persons in the file
void search_by_firstname( char *name);// print out person if in list
void append_file(PERSON *inrecord);// appends a new person to the file

int main( void){
   PERSON ppost;


   return(0);

}
```