Universite Mohammed VI Polytechnique **Year 2020-2021**
Licence LSD **Data Structures**

PRACTICE SESSION
SESSION III

Exercice 1. *A palindrome is a string that reads the same backward as forward, for example strings "z", "aaa", "aba", "abccba" are palindromes, but strings "datascience", "reality", "ab" are not.*
In this exercise you will implement a program that take a string from the user (using the command line arguments) and check if it's a Palindrome using two data structures arrays and a linked list, then we will benchmark.

1. *Get the string to evaluate from the user using the command line arguments.*
2. *Convert the string to a linked list.*
3. *Create a function that take a list of characters and return* `true` *if the string in the linked list is a* `Palindrome` *otherwise* `false`*.*
4. *Calculate the complexity of your function.*
5. *Now, find another way to get the check Palindrome in O(n / 2).*

Exercice 2. *Linked list:*

1. *Create 2 linked list with random values (100 nodes, 30 nodes).*
2. *Create a function called* `sort_list` *then use this function to sort the two linked lists.*
3. *Create a function called* `merge_lists`*, to merge your linked lists into one list.*
4. *Create a function called remove duplicated, then use this function to remove duplicated values from the resulted list.*
5. *Print the result.*
6. *Recreate the same exercise using arrays instead of lists, and compare the time of execution of the two data structures.*

Exercice 3. *Stack ADT:*

In this exercise, we will implement the stack data structure, most used functions.

1. *Using the header file 'stack.h', implement the following functions:*
   - *new_stack: Allocate, fill and return a new stack*
   - *is_empty: Check if the stack is empty*
   - *push: Push a new node to the of the stack*
   - *pop: Remove a node from the top of the stack, if the stack is empty return INT_MIN, otherwise return the popped element.*
   - *peek_stack: Return the top element in the stack, if the stack is empty return INT_MIN.*
   - *print_stack: Print all of the stack elements*

PS: *Create the function using the same prototype in the header file.*

Date: April 9, 2021.

Exercice 4. *Reverse polish notation calculator:*

1. *Using the stack implementation from the previous exercice, write a program that takes a string which contains an equation written in Reverse Polish notation (RPN) as its first argument, evaluates the equation, and prints the result on the standard output followed by a newline.*
   Reverse Polish Notation is a mathematical notation in which every operator follows all of its operands. In RPN, every operator encountered evaluates the previous 2 operands, and the result of this operation then becomes the first of the two operands for the subsequent operator. Operands and operators must be spaced by at least one space.

   You must implement the following operators : "+", "-", "", "/", and "%".*

   If the string isn't valid or there isn't exactly one argument, you must print "Error" on the standard output followed by a newline.

   All the given operands must fit in a "int".

   Examples of formulas converted in RPN:
   3 + 4 == 3 4 +
   *((1 * 2) * 3) - 4 == 1 2 * 3 * 4 - OR 3 1 2 * * 4 -*
   *50 * (5 - (10 / 9)) == 5 10 9 / - 50 **

   Here's how to evaluate a formula in RPN:

   *1 2 * 3 * 4 -*
   *2 3 * 4 -*
   6 4 -
   2

   Or:

   *3 1 2 * * 4 -*
   *3 2 * 4 -*
   6 4 -
   2

   Examples:
   *$ ./rpn_calc "12 * 3 * 4 +" | cat -e*
   10$
   $ ./rpn_calc "1234 +" | cat -e
   Error$
   $ ./rpn_calc | cat -e
   Error$

PS: Lookup the atoi function and it's usage as it will be useful in solving this exercise.