Elementary programming

push_swap

Astek in charge astek_resp@epitech.eu

Abstract: This document is the subject of the push_swap Elementary programming subject

Contents

I Administrative details 2

II Description of the game 3

III Examples 4

IV The program 5

V Options 6

VI Allowed functions 7

Chapter I

Administrative details

- Turn-in directory : Your sources must be turned in on the 2013_CPE_Pushswap repository.
- Your Makefile being at the root of the repository.

Be careful, the norm will be checked on every file you turn in with your lib my !

Chapter II

Description of the game

The game is constituted of two lists, named l_a and l_b. At the start, l_b is empty and l_a contains a certain amount of positive or negative unique numbers. The goal of the game is to make it so l_a contains the same numbers, but in ascending order. To do so, we only have the following operations :

- `sa` : swaps the first 2 elements of l_a
  (does nothing if not enough elements)
- `sb` : swaps the first 2 elements of l_b
  (does nothing if not enough elements)
- `ss` : sa and sb at the same time
- `pa` : takes the first element of l_b and puts it in first position in l_a
  (if l_b is empty, does nothing)
- `pb` : takes the first element of l_a and puts it in first position in l_b
  (if l_a is empty, does nothing)
- `ra` : rotates l_a
  (towards the start, first element becomes last)
- `rb` : rotates l_b
  (towards the start, first element becomes last)
- `rr` : ra and rb at the same time
- `rra` : rotates l_a
  (towards the end, last element becomes the first)
- `rrb` : rotates l_b
  (towards the end, last element becomes the first)
- `rrr` : rra and rrb at the same time

Chapter III

Examples

- Lists a and b will be as follows :
  `l_a` 2 1 3 6 5 8
  `l_b`

- sa
  `l_a` 1 2 3 6 5 8
  `l_b`

- pb pb pb
  `l_a` 6 5 8
  `l_b` 3 2 1

- ra rb (or simply rr)
  `l_a` 5 8 6
  `l_b` 2 1 3

- rra rrb (or simply rrr)
  `l_a` 6 5 8
  `l_b` 3 2 1

- sa
  `l_a` 5 6 8
  `l_b` 3 2 1

- pa pa pa
  `l_a` 1 2 3 5 6 8
  `l_b`

Chapter IV

The program

You must make a program that takes the l_a list as parameter, as a list of parameters (No doubles, all numbers are valid and fit in an integer). The program must display the series of operations allowing to sort the list. Operations are displayed separated by a space, no space at the start or the end, and with a newline at the end. The goal is to sort the list with the fewest operations possible.

```
1   $./push_swap 2 1 3 6 5 8
2   sa pb pb pb sa pa pa pa
3   $
```

Chapter V

Options

You can make the options :

- -v : displays the states of l_a and l_b at each step.
- -vt : the same, in termcaps

Chapter VI

Allowed functions

- write
- malloc
- free
- exit