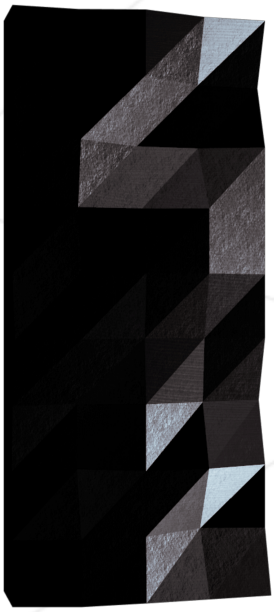

Proiect Algo I

push_swap

Staff Academy+Plus contact@academyplus.ro

Sumar: Cu-cu, vrei sa-mi vezi lista?

Cuprins

Capitolul I

Preambul

Hello world!

- C

```
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
    return 0;
}
```

- ASM

```
cseg segment
assume cs:cseg, ds:cseg
org 100h
main proc
jmp debut
mess db 'Hello world!$'
debut:
mov dx, offset mess
mov ah, 9
int 21h
ret
main endp
cseg ends
end main
```

- LOLCODE

```
HAI
```

```
  CAN HAS STDIO?
  VISIBLE "HELLO WORLD!"
  KTHXBYE
```

- PHP

```
<?php
 echo "Hello world!";
 ?>
```

- BrainFuck

```
   ++++++++++[>+++++++>++++++++++>+++>+<<<<-]
   >++.>+.+++++++..+++.>++.
   <<+++++++++++++++.>.+++.------.--------.>+.>.
```

- C#

```
using System;

public class HelloWorld {
   public static void Main () {
      Console.WriteLine("Hello world!");
   }
}
```

- HTML5

```
<!DOCTYPE html>
<html>
   <head>
       <meta charset="utf-8">
       <title>Hello world !</title>
   </head>
   <body>
       <p>Hello World !</p>
   </body>
</html>
```

Capitolul II

Descrierea jocului

- Jocul consta din 2 liste numite l_a si l_b.
- La inceput l_b este goala si l_a contine un anumit numar de numere pozitive sau negative (fara a le dubla).
- Obiectivul jocului este de a face ca, la iesire, lista l_a sa contina aceleasi numere, dar in ordine crescatoare.
- Pentru a realiza asta, nu dispuneti decat de operatiile urmatoare:
  - `sa` : swap pe primele 2 elemente ale listei l_a
    (nu are importanta daca e doar unul, sau niciun element).
  - `sb` : swap pe primele 2 elemente ale listei l_b
    (nu are importanta daca e doar unul, sau niciun element).
  - `ss` : sa si sb in acelasi timp.
  - `pa` : ia primul element al listei l_b si il pune pe prima pozitie in lista l_a.
    (nu conteaza daca lista l_b e goala).
  - `pb` : ia primul element al listei l_a si il pune pe prima pozitie in lista l_b.
    (nu conteaza daca lista l_a e goala).
  - `ra` : roteste l_a
    (spre inceputul listei, primul element devine ultimul).
  - `rb` : roteste l_b
    (spre inceputul listei, primul element devine ultimul).
  - `rr` : ra si rb in acelasi timp.
  - `rra` : roteste l_a
    (spre sfarsitul listei, ultimul element devine primul).
  - `rrb` : roteste l_b
    (spre sfarsitul listei, ultimul element devine primul).

- `rrr` : rra si rrb in acelasi timp.

Capitolul III

Exemple

- Listele a si b vor fi definite astfel:
  `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 (se poate spune de asemenea rr)
  `l_a` 5 8 6
  `l_b` 2 1 3

- rra rrb (se poate spune de asemenea 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`

Capitolul IV

Subiect

- Trebuie sa creati un program ce ia ca parametru lista l_a sub forma unei liste de parametri (Fara a-i dubla, toate numerele sunt bune si returneaza un intreg).
- Programul trebuie sa afiseze seria de operatii (operatori) ce permite ordonarea elementelor listei. Operatorii vor fi afisati, separati de un spatiu, fara niciun spatiu la inceput ori la sfarsit, totul urmat de '\n'.
- Scopul acestui program este ordonarea listei cu cel mai mic numar de operatii posibil.

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

In caz de eroare, veti afisa "Error" urmat de '\n' la iesirea de erori.

Capitolul V

Subiect - Parte bonus

Bonusul va fi evaluat doar daca partea obligatorie este completa. Prin completa, se intelege ca ea este cvasi-terminata. O nota de cel putin 18 puncte va ofera sansa ca partea bonus sa fie evaluata.

Iata cateva idei de bonus interesante de realizat si chiar utile. Puteti de asemenea sa adaugati functionalitati bonus la initiativa voastra, ce vor fi punctate la discretia evaluatorului vostru.

- Pentru debug optiunile: -v poate afisa starea listei la fiecare etapa, -c poate afisa colorat ultima actiune etc.

Capitolul VI

Instructiuni

- Acest proiect trebuie sa respecte constrangerile listate mai jos.
- Programul trebuie sa se numeasca `push_swap`.
- Trebuie sa aveti un fisier Makefile.
- Proiectul trebuie scris conform standardului de cod (Norme).
- Trebuie sa tratati erorile in mod eficient. In niciun caz programul nu trebuie sa se termine intr-un mod neasteptat (Eroare de segmentare etc...)
- Trebuie sa livrati, in radacina directorului vostru de lucru/livrare, un fisier `auteur` continand login-ul vostru urmat de '\n':

```
$>cat -e auteur
xlogin ylogin$
$>
```

- Aveti dreptul sa folositi urmatoarele functii:
  - write
  - malloc
  - free
  - exit
- Puteti adresa intrebari pe forum, pe jabber, IRC, ...

Capitolul VII

Notare

- Notarea programului `push_swap` se face in doi timpi:
  - In primul rand, partea obligatorie va fi testata. Ea va fi notata cu maxim 10 puncte.
  - Apoi va fi supusa unei incercari calitatea algoritmului vostru, calitate care va fi notata cu maxim 10 puncte.
  - In sfarsit, bonusul va fi evaluat. El va fi notat cu maxim 5 puncte.
    - Acestea insa vor fi evaluate doar daca partea obligatorie a fost realizata integral. Consideram ca doar nota 18 va fi suficienta pentru evaluarea partii bonus.
    - De asemenea, optimizarea calitatii anumitor elemente de cod vor fi evaluate, putand obtine astfel puncte suplimentare la partea de bonus.
- Succes tuturor!