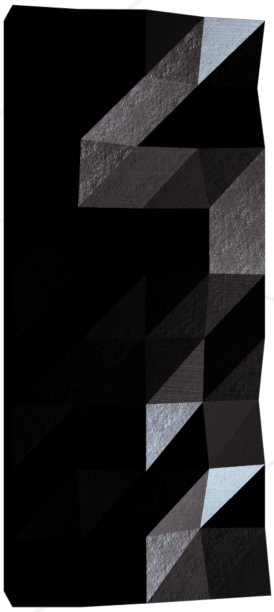

Projet Algo I

push_swap

42 staff staff@42.fr

Résumé: Coucou, tu veux voir ma list ?

Table des matières

Chapitre I

Préambule

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>
```

Chapitre II

Description du jeu

- Le jeu est constitué de 2 listes nommées l_a et l_b.
- Au départ l_b est vide et l_a contient un certain nombre de nombres positifs ou négatifs (sans doublons).
- Le but du jeu est de faire en sorte que l_a contienne les mêmes nombres mais dans l'ordre croissant.
- Pour ce faire, on ne dispose que des opérations suivantes :
  - `sa` : swap les 2 premiers éléments de l_a
    (ne fait rien s'il n'y en a qu'un ou aucun).
  - `sb` : swap les 2 premiers éléments de l_b
    (ne fait rien s'il n'y en a qu'un ou aucun).
  - `ss` : sa et sb en même temps.
  - `pa` : prend le premier élément de l_b et le met en premier dans l_a.
    (ne fait rien si l_b est vide).
  - `pb` : prend le premier élément de l_a et le met en premier dans l_b.
    (ne fait rien si l_a est vide).
  - `ra` : rotate l_a
    (vers le début, le premier élément devient le dernier).
  - `rb` : rotate l_b
    (vers le debut, le premier élément devient le dernier).
  - `rr` : ra et rb en meme temps.
  - `rra` : rotate l_a
    (vers la fin, le dernier élément devient le premier).
  - `rrb` : rotate l_b
    (vers la fin, le dernier élément devient le premier).
  - `rrr` : rra et rrb en même temps.

Chapitre III

Exemples

- La liste a et b sera defini ainsi :
  `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 (on peut donc aussi dire rr)
  `l_a` 5 8 6
  `l_b` 2 1 3

- rra rrb (on peut donc aussi dire 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`

Chapitre IV

Sujet

- Vous devez faire un programme qui prend en paramètre la liste l_a sous la forme d'une liste de paramètres (Pas de doublons, tout les nombres sont bons et rentrent dans un entier).
- Le programme doit afficher la suite d'opérations qui permet de trier la liste. Les operations seront affichées separées par un espace, pas d'espace au debut ni à la fin, le tout suivi d'un '\n'.
- Le but est de trier la liste avec le moins d'opérations possibles.

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

En cas d'erreur, vous afficherez "Error" suivi d'un '\n' sur la sortie d'erreur.

Chapitre V

Sujet - Partie bonus

Les bonus ne seront évalués que si votre partie obligatoire est complète. Par complète, on entend qu'elle est, au moins, quasiment entièrement réalisée. Une note de 18 ou plus offrira la possibilité de se faire noter les bonus.

Voici quelques idées de bonus intéressants à réaliser, voire même utiles. Vous pouvez évidemment ajouter des bonus de votre invention, qui seront évalués à la discrétion de vos correcteurs.

- Ajout d'options pour le debug : -v peut afficher l'etat des listes a chaque coup, -c peut faire de afficher en couleur la dernière action, etc.

Chapitre VI

Consignes

- Ce projet doit respecter les contraintes listées ici.
- La programme doit s'appeller `push_swap`.
- Vous devez avoir un Makefile.
- Votre projet doit être à la Norme.
- Vous devez gérer les erreurs de façon sensible. En aucun cas votre programme ne doit quitter de façon inattendue (Segmentation fault, etc...)
- Vous devez rendre, à la racine de votre dépôt de rendu, un fichier `auteur` contenant votre login suivi d'un '\n' :

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

- Vous avez le droit d'utiliser les fonctions suivantes :
  - write
  - malloc
  - free
  - exit
- Vous pouvez poser vos questions sur le forum, sur jabber, IRC, ...

Chapitre VII

Notation

- La notation de `push_swap` s'effectue en deux temps :
  - En premier lieu, votre partie obligatoire sera testée. Elle sera notée sur 10 points.
  - Ensuite, la qualité de votre algorithme, qui sera notée sur 10, sera mis à l'epreuve dans une compétition.
  - Enfin, vos bonus seront évalués. Ils seront notés sur 5 points.
    - Ils ne seront évalués que si votre partie obligatoire est réalisée completement. Nous considerons que la note de 18 est suffisante pour compatibliser les bonus.
    - Également, l'optimisation de la qualité de certains éléments de votre code seront évalués et pourront donner lieu à des points supplémentaires dans cette partie bonus.
- Bon courage à tous !