LVL 0 : 0-0-aff_a

Assignment name : aff_aExpected files : aff_a.c
Allowed functions: write

Écrire un programme qui prend une chaîne de caractères et affiche le premiercaractère 'a' qu'il y rencontre, suivi d'un newline. S'il n'y a pas decaractère 'a' dans la chaîne, le programme affiche juste un newline.

Si lenombre de paramètres n'est pas 1, le programme affiche 'a' suivi d'un newline.

Exemple:$> ./aff_a "abc" | cat -e
a$
$> ./aff_a "dubO a POIL" | cat -e
a$
$> ./aff_a "zz sent le poney" | cat -e
$
$> ./aff_a | cat -e
a$

```c
/* ************************************************************************** */
/*                                                                            */
/*                                                        :::      ::::::::   */
/*   maff_alpha.c                                       :+:      :+:    :+:   */
/*                                                    +:+ +:+         +:+     */
/*   By: nbeny <marvin@42.fr>                       +#+  +:+       +#+        */
/*                                                +#+#+#+#+#+   +#+           */
/*   Created: 2016/07/21 18:15:33 by nbeny             #+#    #+#             */
/*   Updated: 2016/07/21 18:24:12 by nbeny            ###   ########.fr       */
/*                                                                            */
/* ************************************************************************** */

#include <unistd.h>

void    ft_putchar(char c)
{
    write(1, &c, 1);
}

void    ft_alphabet(void)
{
    int i;
    int poney;

    poney = 'B';
    i = 'a';
    while (i <= 'y' && poney <= 'Z')
    {
        ft_putchar(i);
        ft_putchar(poney);
        i = i + 2;
        poney = poney + 2;
    }
    ft_putchar('\n');
}

int     main(void)
{
    ft_alphabet();
    return (0);
}
```

LVL 0 : 0-0-ft_countdown

```
Assignment name  : ft_countdown
Expected files   : ft_countdown.c
Allowed functions: write
--------------------------------------------------------------------------------

Écrire un programme qui affiche tous les chiffres en ordre descendant, suivis
d'un newline.

Exemple:
$> ./ft_countdown | cat -e
9876543210$
$>
```

```c
/* ************************************************************************** */
/*                                                                            */
/*                                                        :::      ::::::::   */
/*   ft_countdown.c                                     :+:      :+:    :+:   */
/*                                                    +:+ +:+         +:+     */
/*   By: recharif <marvin@42.fr>                    +#+  +:+       +#+        */
/*                                                +#+#+#+#+#+   +#+           */
/*   Created: 2016/07/27 11:05:21 by recharif          #+#    #+#             */
/*   Updated: 2016/07/27 11:06:37 by recharif         ###   ########.fr       */
/*                                                                            */
/* ************************************************************************** */

#include <unistd.h>

int     main(void)
{
    write(1, "9876543210", 10);
    write(1, "\n", 1);
    return (0);
}
```

LVL 0 : 0-0-ft_print_numbers

```
Assignment name  : ft_print_numbers
Expected files   : ft_print_numbers.c
Allowed functions: write
--------------------------------------------------------------------------------

Écrire une fonction qui affiche tous les chiffres dans l'ordre croissant.

Elle devra être prototypée de la façon suivante :

void    ft_print_numbers(void);
```

```c
/* ************************************************************************** */
/*                                                                            */
/*                                                        :::      ::::::::   */
/*   ft_print_numbers.c                                 :+:      :+:    :+:   */
/*                                                    +:+ +:+         +:+     */
/*   By: nbeny <marvin@42.fr>                       +#+  +:+       +#+        */
/*                                                +#+#+#+#+#+   +#+           */
/*   Created: 2016/07/21 17:42:08 by nbeny             #+#    #+#             */
/*   Updated: 2016/07/21 17:50:55 by nbeny            ###   ########.fr       */
/*                                                                            */
/* ************************************************************************** */

#include <unistd.h>

void    ft_putchar(char c)
{
    write(1, &c, 1);
}

void    ft_print_numbers(void)
{
    int i;

    i = '0';
    while (i <= '9')
    {
        ft_putchar(i);
        i++;
    }
}
```

LVL 0 : 0-0-maff_alpha

```
Assignment name  : maff_alpha
Expected files   : maff_alpha.c
Allowed functions: write
--------------------------------------------------------------------------------

Écrire un programme qui affiche l'alphabet une lettre sur 2 en minuscule, et le
reste en majuscule (Voir l'exemple), suivi d'un '\n'.

Exemple:

$> ./maff_alpha | cat -e
aBcDeFgHiJkLmNoPqRsTuVwXyZ$
```

```c
/* ************************************************************************** */
/*                                                                            */
/*                                                        :::      ::::::::   */
/*   maff_alpha.c                                       :+:      :+:    :+:   */
/*                                                    +:+ +:+         +:+     */
/*   By: exam <marvin@42.fr>                        +#+  +:+       +#+        */
/*                                                +#+#+#+#+#+   +#+           */
/*   Created: 2016/07/15 18:08:05 by exam             #+#    #+#             */
/*   Updated: 2016/07/15 18:10:41 by exam            ###   ########.fr       */
/*                                                                            */
/* ************************************************************************** */

#include <unistd.h>

int     main(void)
{
        write(1, "aBcDeFgHiJkLmNoPqRsTuVwXyZ\n", 27);
            return(0);
}
```

LVL 0 : 0-1-aff_last_param

```
Assignment name  : aff_last_param
Expected files   : aff_last_param.c
Allowed functions: write
--------------------------------------------------------------------------------

Écrire un programme qui prend des chaînes, et affiche son dernier argument suivi d'un
newline.

S'il y a moins d'un argument, le programme affiche juste un newline.

Exemples:

$> ./aff_last_param "zaz" "mange" "des" "chats" | cat -e
chats$
$> ./aff_last_param "j'aime le savon" | cat -e
j'aime le savon$
$> ./aff_last_param
$
```

```c
/* ************************************************************************** */
/*                                                                            */
/*                                                        :::      ::::::::   */
/*   main.c                                             :+:      :+:    :+:   */
/*                                                    +:+ +:+         +:+     */
/*   By: jpeguet <marvin@42.fr>                     +#+  +:+       +#+        */
/*                                                +#+#+#+#+#+   +#+           */
/*   Created: 2016/07/22 12:07:07 by jpeguet           #+#    #+#             */
/*   Updated: 2016/07/22 12:34:51 by jpeguet          ###   ########.fr       */
/*                                                                            */
/* ************************************************************************** */

#include <unistd.h>

void    ft_putchar(char c)
{
    write(1, &c, 1);
}

void    ft_putstr(char *str)
{
    int     i;

    i = 0;
    while (str[i] != '\0')
    {
        ft_putchar(str[i]);
        i++;
    }
}

int     main(int ac, char **av)
{
    int     i;

    i = 0;
    while (i != (ac - 1))
    {
        i++;
    }
    ft_putstr(av[i]);
    ft_putchar('\n');
    return(0);
}
```

LVL 0 : 0-1-only_z

```
Assignment name  : only_z
Expected files   : only_z.c
Allowed functions: write
--------------------------------------------------------------------------------

Écrire un programme qui affiche un caractère 'z' sur la sortie standard.
```

```
/* ************************************************************************** */
/*                                                                            */
/*                                                        :::      ::::::::   */
/*   only_z.c                                           :+:      :+:    :+:   */
/*                                                    +:+ +:+         +:+     */
/*   By: amarteau <marvin@42.fr>                    +#+  +:+       +#+        */
/*                                                +#+#+#+#+#+   +#+           */
/*   Created: 2016/07/22 12:54:00 by amarteau          #+#    #+#             */
/*   Updated: 2016/07/22 12:56:23 by amarteau         ###   ########.fr       */
/*                                                                            */
/* ************************************************************************** */

#include <unistd.h>

int main()
{
    write(1, "z", 1);
    return (0);
}
```

LVL 0 : 0-2-aff_z

```
Assignment name  : aff_z
Expected files   : aff_z.c
Allowed functions: write
--------------------------------------------------------------------------------

Écrire un programme qui prend en paramètre une chaîne de caractères, et qui
affiche sur la sortie standard le premier caractère 'z' rencontré dans
cette chaîne, suivi de '\n'. Si aucun 'z'
n'est rencontré dans la chaîne, le programme affiche 'z' suivi de
'\n'. Si le nombre de paramètres est différent de 1, le
programme affiche 'z' suivi de '\n'.

Exemple:

$> ./aff_z "abc" | cat -e
z$
$> ./aff_z "dubO a POIL" | cat -e
z$
$> ./aff_z "zaz sent le poney" | cat -e
z$
$> ./aff_z | cat -e
z$
```

```
/* ************************************************************************** */
/*                                                                            */
/*                                                        :::      ::::::::   */
/*   aff_z.c                                            :+:      :+:    :+:   */
/*                                                    +:+ +:+         +:+     */
/*   By: amarteau <marvin@42.fr>                    +#+  +:+       +#+        */
/*                                                +#+#+#+#+#+   +#+           */
/*   Created: 2016/07/22 13:21:55 by amarteau          #+#    #+#             */
/*   Updated: 2016/07/22 13:29:48 by amarteau         ###   ########.fr       */
/*                                                                            */
/* ************************************************************************** */

#include <unistd.h>

int main(int ac,char **av)
{
    if (ac)
    {
        **av = 'z';
        write(1, *av,1);
        write(1 , "\n", 1);
    }
    return (0);
}
```

LVL 1 : 1-0-ft_strcpy

```
Assignment name  : ft_strcpy
Expected files   : ft_strcpy.c
Allowed functions:
--------------------------------------------------------------------------------

Reproduisez à l'identique le comportement de la fonction strcpy (man strcpy).

Votre fonction devra être prototypée de la façon suivante :

char    *ft_strcpy(char *s1, char *s2);
```

```c
/* ************************************************************************** */
/*                                                                            */
/*                                                        :::      ::::::::   */
/*   ft_strcpy.c                                        :+:      :+:    :+:   */
/*                                                    +:+ +:+         +:+     */
/*   By: rcargou <rcargou@student.42.fr>            +#+  +:+       +#+        */
/*                                                +#+#+#+#+#+   +#+           */
/*   Created: 2014/11/03 15:36:35 by rcargou           #+#    #+#             */
/*   Updated: 2015/09/02 15:42:41 by rcargou          ###   ########.fr       */
/*                                                                            */
/* ************************************************************************** */

char    *ft_strcpy(char *dst, const char *src)
{
    int a;

    a = 0;
    while (src[a])
    {
        dst[a] = src[a];
        a++;
    }
    dst[a] = '\0';
    return (dst);
}
```

LVL 1 : 1-0-ft_strlen

```
Assignment name  : ft_strlen
Expected files   : ft_strlen.c
Allowed functions:
--------------------------------------------------------------------------------

Écrire une fonction qui renvoie la longueur d'une chaîne de caractères.

Elle devra être prototypée de la façon suivante :

int ft_strlen(char *str);
```

```c
/* ************************************************************************** */
/*                                                                            */
/*                                                        :::      ::::::::   */
/*   ft_strlen.c                                        :+:      :+:    :+:   */
/*                                                    +:+ +:+         +:+     */
/*   By: nbeny <marvin@42.fr>                       +#+  +:+       +#+        */
/*                                                +#+#+#+#+#+   +#+           */
/*   Created: 2016/07/21 18:51:25 by nbeny             #+#    #+#             */
/*   Updated: 2016/07/21 18:56:42 by nbeny            ###   ########.fr       */
/*                                                                            */
/* ************************************************************************** */

int     ft_strlen(char *str)
{
    int i;

    i = 0;
    while (str[i])
        i++;
    return (i);
}
```

LVL 1 : 1-0-ft_swap

```
Assignment name  : ft_swap
Expected files   : ft_swap.c
Allowed functions:
--------------------------------------------------------------------------------

Écrire une fonction qui échange le contenu de deux entiers dont les adresses
sont passées en paramêtres.

Elle devra être prototypée de la façon suivante :

void    ft_swap(int *a, int *b);
```

```c
/* ************************************************************************** */
/*                                                                            */
/*                                                        :::      ::::::::   */
/*   ft_swap.c                                          :+:      :+:    :+:   */
/*                                                    +:+ +:+         +:+     */
/*   By: jpeguet <marvin@42.fr>                     +#+  +:+       +#+        */
/*                                                +#+#+#+#+#+   +#+           */
/*   Created: 2016/07/07 22:17:01 by jpeguet           #+#    #+#             */
/*   Updated: 2016/07/07 22:20:51 by jpeguet          ###   ########.fr       */
/*                                                                            */
/* ************************************************************************** */

void    ft_swap(int *a, int *b)
{
    *a = *a + *b;
    *b = *a - *b;
    *a = *a - *b;
}
```

Autre solution :

```c
void    ft_swap(int *a, int *b)
{
    int tmp;

    tmp = *a;
    *a = *b;
    *b = tmp;
}
```

LVL 1 : 1-0-repeat_alpha

```
Assignment name  : repeat_alpha
Expected files   : repeat_alpha.c
Allowed functions: write
--------------------------------------------------------------------------------

Écrire un programme repeat_alpha qui prend une chaîne et l'affiche en répétant
chaque caractère alphabétique autant de fois que son index dans l'alphabet,
suivie d'un newline.

'a' devient 'a', 'b' devient 'bb', 'e' devient 'eeeee', etc...

La casse ne change pas.

Si le nombre d'arguments n'est pas 1, affiche juste un newline.

Exemples:

$>./repeat_alpha "abc"
abbccc
$>./repeat_alpha "Alex." | cat -e
Allllllllllleeeeexxxxxxxxxxxxxxxxxxxxxxxx.$
$>./repeat_alpha 'abacadaba 42!' | cat -e
abbacccaddddabba 42!$
$>./repeat_alpha | cat -e
$
$>
$>./repeat_alpha "" | cat -e
$
$>
```

```c
#include <unistd.h>

void    ft_putchar(char c)
{
    write(1, &c, 1);
}

void    ft_repeat(char c, char v)
{
    int     j;

    j = 0;
    while (j <= c - v)
    {
        ft_putchar(c);
        j++;
    }
}

int     main(int ac, char **av)
{
    int     i;
    int     j;

    i = 0;
    if (ac == 2)
    {
        while (av[1][i])
        {
            j = 0;
            if (av[1][i] >= 'a' && av[1][i] <= 'z')
                ft_repeat(av[1][i], 'a');
            else if (av[1][i] >= 'A' && av[1][i] <= 'Z')
                ft_repeat(av[1][i], 'A');
            else
                ft_putchar(av[1][i]);
            i++;
        }
    }
    ft_putchar('\n');
    return (0);
}
```

LVL 1 : 1-0-rev__print

```
Assignment name  : rev_print
Expected files   : rev_print.c
Allowed functions: write
--------------------------------------------------------------------------------

Écrire un programme qui prend une chaîne et l'affiche en ordre inverse suivie
d'un newline.

Si le nombre d'arguments n'est pas 1, le programme affiche un newline.

Exemples:

$> ./rev_print "zaz" | cat -e
zaz$
$> ./rev_print "dub0 a POIL" | cat -e
LIOP a 0bud$
$> ./rev_print | cat -e
$
```

```c
/* ************************************************************************** */
/*                                                                            */
/*                                                        :::      ::::::::   */
/*   rev_print.c                                        :+:      :+:    :+:   */
/*                                                    +:+ +:+         +:+     */
/*   By: amarteau <marvin@42.fr>                    +#+  +:+       +#+        */
/*                                                +#+#+#+#+#+   +#+           */
/*   Created: 2016/07/22 14:41:19 by amarteau          #+#    #+#             */
/*   Updated: 2016/07/22 15:00:41 by amarteau         ###   ########.fr       */
/*                                                                            */
/* ************************************************************************** */

#include <unistd.h>

int ft_putchar(char c)
{
    write(1, &c ,1);
    return (0);
}


int main(int ac, char **av)
{
    int i;

    i = 0;
    if (ac == 2)
    {
        while (av[1][i])
            i++;
        while (i >= 0)
        {
            ft_putchar(av[1][i]);
            i--;
        }
    }
    ft_putchar('\n');
}
```

LVL 1 : 1-0-search_and_replace

```
Assignment name  : search_and_replace
Expected files   : search_and_replace.c
Allowed functions: write, exit
--------------------------------------------------------------------------------

Écrire un programme search_and_replace qui prend 3 arguments, le premier est
une chaîne dans laquelle on veut remplacer une lettre (2ème argument) par une
autre (3ème argument).

Si le nombre d'arguments n'est pas 3, affiche juste un newline.

Si le deuxième argument n'est pas contenu dans la chaîne, alors le programme ne
change rien à la chaîne et l'affiche telle quelle suivie d'un newline.

Exemples:
$>./search_and_replace "Papache est un sabre" "a" "o"
Popoche est un sobre
$>./search_and_replace "zaz" "art" "zul" | cat -e
$
$>./search_and_replace "zaz" "r" "u" | cat -e
zaz$
$>./search_and_replace "jacob" "a" "b" "c" "e" | cat -e
$
$>./search_and_replace "ZoZ eT Dovid oiME le METol." "o" "a" | cat -e
ZaZ eT David aiME le METal.$
$>./search_and_replace "wNcOre Un ExEmPle Pas Facilw a Ecrirw " "w" "e" | cat -e
eNcOre Un ExEmPle Pas Facile a Ecrire $
```

```c
/* ************************************************************************** */
/*                                                                            */
/*                                                        :::      ::::::::   */
/*   search_and_replace.c                               :+:      :+:    :+:   */
/*                                                    +:+ +:+         +:+     */
/*   By: recharif <marvin@42.fr>                    +#+  +:+       +#+        */
/*                                                +#+#+#+#+#+   +#+           */
/*   Created: 2016/07/27 11:52:36 by recharif          #+#    #+#             */
/*   Updated: 2016/07/27 12:06:06 by recharif         ###   ########.fr       */
/*                                                                            */
/* ************************************************************************** */

#include <unistd.h>

void    ft_putchar(char c)
{
    write(1, &c, 1);
}

int     main(int ac, char **av)
{
    int     i;

    i = 0;
    if (ac == 4 && av[2][1] == 0 && av[3][1] == 0)
    {
        while (av[1][i])
        {
            if (av[1][i] == av[2][0])
                av[1][i] = av[3][0];
            ft_putchar(av[1][i]);
            i++;
        }
    }
    ft_putchar('\n');
    return (0);
}
```

LVL 1 : 1-0-ulstr

```
Assignment name  : ulstr
Expected files   : ulstr.c
Allowed functions: write
--------------------------------------------------------------------------------

Écrire un programme qui prend en paramètre une chaîne de caractères, et qui
transforme toutes ses minuscules en majuscules et toutes ses majuscules en
minuscules. Les autres caractères restent inchangés.

Vous devez afficher le résultat suivi d'un '\n'.

Si le nombre de paramètres transmis est différent de 1, le programme affiche
'\n'.

Exemples :

$>./ulstr "L'eSPrit nE peUt plUs pRogResSer s'Il staGne et sI peRsIsTent VAnIte et auto-
justification." | cat -e
l'EspRIT Ne PEuT PLuS PrOGrESsER S'iL STAgNE ET Si PErSiStENT vaNiTE ET AUTO-JUSTIFICATION.$
$>./ulstr "S'enTOuRer dE sECreT eSt uN sIGnE De mAnQuE De coNNaiSSanCe.  " | cat -e
s'ENtoUrER De SecREt EsT Un SigNe dE MaNqUe dE COnnAIssANcE.  $
$>./ulstr "3:21 Ba  tOut  moUn ki Ka di KE m'en Ka fe fot" | cat -e
3:21 bA  ToUT  MOuN KI kA DI ke M'EN kA FE FOT$
$>./ulstr | cat -e
$
```

```c
/* ************************************************************************** */
/*                                                                            */
/*                                                        :::      ::::::::   */
/*   ulstr.c                                            :+:      :+:    :+:   */
/*                                                    +:+ +:+         +:+     */
/*   By: amarteau <marvin@42.fr>                    +#+  +:+       +#+        */
/*                                                +#+#+#+#+#+   +#+           */
/*   Created: 2016/07/25 10:54:50 by amarteau          #+#    #+#             */
/*   Updated: 2016/07/25 11:28:56 by amarteau         ###   ########.fr       */
/*                                                                            */
/* ************************************************************************** */

#include <unistd.h>

int ft_putchar(char c)
{
    write(1, &c ,1);
    return (0);
}

int main(int argc, char **argv)
{
    char *str;

    if (argc == 2)
    {
        str = argv[1];
        while (*str)
        {
            if (*str >= 'a' && *str <= 'z')
                ft_putchar(*str - 32);
            else if (*str >= 'A' && *str <= 'Z')
                ft_putchar(*str + 32);
            else
                ft_putchar(*str);
            str++;
        }
    }
    ft_putchar('\n');
}
```

LVL 1 : 1-1-rot_13

```
Assignment name  : rot_13
Expected files   : rot_13.c
Allowed functions: write
--------------------------------------------------------------------------------

Écrire un programme nommé rotone, qui prend en paramètre une chaîne de
caractères, et qui affiche cette chaîne en remplaçant chaque caractère
alphabétique par le caractère situé 13 caractères plus loin dans l'ordre
alphabétique.

'z' devient 'm' et 'Z' devient 'M'. Les majuscules restent des majuscules, les
minuscules restent des minuscules.

L'affichage se termine toujours par un retour à la ligne.

Si le nombre de paramètres transmis est différent de 1, le programme affiche
'\n'.

Exemple:

$>./rot_13 "abc"
nop
$>./rot_13 "My horse is Amazing." | cat -e
Zl ubefr vf Nznmvat.$
$>./rot_13 "AkjhZ zLKIJz , 23y " | cat -e
NxwuM mYXVWm , 23l $
$>./rot_13 | cat -e
$
$>
$>./rot_13 "" | cat -e
$
$>
```

```c
/* ************************************************************************** */
/*                                                                            */
/*                                                        :::      ::::::::   */
/*   rot_13.c                                           :+:      :+:    :+:   */
/*                                                    +:+ +:+         +:+     */
/*   By: amarteau <marvin@42.fr>                    +#+  +:+       +#+        */
/*                                                +#+#+#+#+#+   +#+           */
/*   Created: 2016/07/22 16:08:02 by amarteau          #+#    #+#             */
/*   Updated: 2016/07/22 16:35:44 by amarteau         ###   ########.fr       */
/*                                                                            */
/* ************************************************************************** */

#include <unistd.h>

int ft_putchar(char c)
{
    write(1, &c ,1);
    return(0);
}

int main(int ac, char **av)
{
    int i;

    i = 0;
    if (ac == 2)
    {
        while(av[1][i])
        {
            if (av[1][i] >= 'a' && av[1][i] <= 'z')
                av[1][i] = (av[1][i] - 'a' + 13)%26 +'a';
            else if (av[1][i] >= 'A' && av[1][i] <= 'Z')
                av[1][i] = (av[1][i] - 'A' + 13)%26 +'A';
            ft_putchar(av[1][i]);
            i++;
        }
    }
    ft_putchar('\n');
}
```

LVL 1 : 1-1-rotone

```
Assignment name  : rotone
Expected files   : rotone.c
Allowed functions: write
--------------------------------------------------------------------------------

Écrire un programme nommé rotone, qui prend en paramètre une chaîne de
caractères, et qui affiche cette chaîne en remplaçant chaque caractère
alphabétique par le caractère suivant dans l'ordre alphabétique.

'z' devient 'a' et 'Z' devient 'A'. Les majuscules restent des majuscules, les
minuscules restent des minuscules.

L'affichage se termine toujours par un retour à la ligne.

Si le nombre de paramètres transmis est différent de 1, le programme affiche
'\n'.

Exemple:

$>./rotone "abc"
bcd
$>./rotone "Les stagiaires du staff ne sentent pas toujours tres bon." | cat -e
Mft tubhjbjsft ev tubgg of tfoufou qbt upvkpvst usft cpo.$
$>./rotone "AkjhZ zLKIJz , 23y " | cat -e
BlkiA aMLJKa , 23z $
$>./rotone | cat -e
$
$>
$>./rotone "" | cat -e
$
$>
```

```c
#include <unistd.h>

void    ft_putchar(char c)
{
    write(1, &c, 1);
}

void    ft_rotone(char *str)
{
    int     i;

    i = 0;
    while (str[i])
    {
        if ((str[i] >= 'a' && str[i] < 'z') || (str[i] >= 'A' && str[i] < 'Z'))
            ft_putchar(str[i] + 1);
        else if (str[i] == 'z')
            ft_putchar('a');
        else if (str[i] == 'Z')
            ft_putchar('A');
        else
            ft_putchar(str[i]);
        i++;
    }
}

int     main(int argc, char **argv)
{
    if (argc == 2)
    {
        ft_rotone(argv[1]);
        ft_putchar('\n');
    }
    else
        ft_putchar('\n');
    return (0);
}
```

LVL 1 : 1-2-first_word

```
Assignment name  : first_word
Expected files   : first_word.c
Allowed functions: write
--------------------------------------------------------------------------------

Écrire un programme qui prend une chaîne de caractères en paramètre, et qui
affiche le premier mot de cette chaîne, suivi d'un '\n'.

On appelle "mot" une portion de chaîne de caractères délimitée soit par des
espaces et/ou des tabulations, soit par le début / fin de la chaîne.

Si le nombre de paramètres transmis est différent de 1, ou s'il n'y a aucun mot
à afficher, le programme affiche '\n'.

Exemple:

$> ./first_word "FOR PONY" | cat -e
FOR$
$> ./first_word "this        ...       is sparta, then again, maybe    not" | cat -e
this$
$> ./first_word "   " | cat -e
$
$> ./first_word "a" "b" | cat -e
$
$> ./first_word "  lorem,ipsum  " | cat -e
lorem,ipsum$
$>
```

```c
#include <unistd.h>

int ft_putchar(char c)
{
    write(1, &c ,1);
    return (0);
}

int main(int ac, char **av)
{
    int i;

    i = 0;
    if (ac == 2)
    {
            while (av[1][i] == ' ' || av[1][i] == '\t')
                i++;
            while(av[1][i] && av[1][i] != ' ' && av[1][i] != '\t')
                {
                    ft_putchar(av[1][i]);
                    i++;
                }
    }
    ft_putchar('\n');
}
```

LVL 1 : 1-2-ft_putstr

```
Assignment name  : ft_putstr
Expected files   : ft_putstr.c
Allowed functions: write
--------------------------------------------------------------------------------

Écrire une fonction qui affiche une chaîne de caractères sur la sortie standard.

Le pointeur passé à la fonction est l'adresse du premier caractère de la chaîne.

Elle devra être prototypée de la façon suivante :

void    ft_putstr(char *str);
```

```c
#include <unistd.h>

int     ft_putchar(char c)
{
    write(1, &c, 1);
    return (0);
}

void ft_putstr(char *str)
{
    int i;

    i = 0;
    while (str[i])
    {
        ft_putchar(str[i]);
        i++;
    }
}
```

LVL 2 : 2-2-ft_atoi

```
Assignment name  : ft_atoi
Expected files   : ft_atoi.c
Allowed functions: None
--------------------------------------------------------------------------------

Écrire une fonction qui convertit une chaîne en un entier (type int) et le retourne.

Marche comme la fonction standard atoi(const char *str), voir le man.

La fonction doit être prototypée comme suit:

int ft_atoi(const char *str);
```

```c
int ft_atoi(char *str)
{
    int i;
    int res;
    int flag;

    i = 0;
    res = 0;
    flag = 0;
    while (str[i] == ' ' || str[i] == '\t')
        i++;
    if (str[i] == '+' || str[i] == '-')
    {
        if (str[i] == '-')
            flag = 1;
        i++;
    }
    while (str[i] >= '0' && str[i] <= '9')
    {
        res *= 10;
        res += str[i] - '0';
        i++;
    }
    if (flag)
        res = -res;
    return (res);
}
```

LVL 2 : 2-0-ft_strdup

```
Assignment name  : ft_strdup
Expected files   : ft_strdup.c
Allowed functions: malloc
--------------------------------------------------------------------------------

Reproduisez à l'identique le comportement de la fonction strdup (man strdup).

Votre fonction devra être prototypée de la façon suivante :

char    *ft_strdup(char *src);
```

```c
#include <stdlib.h>
#include "libft.h"

char    *ft_strdup(const char *s1)
{
    char    *dest;
    int     a;

    a = 0;
    dest = malloc(sizeof(char) * ft_strlen(s1) + 1);
    if (dest)
    {
        while (s1[a])
        {
            dest[a] = s1[a];
            a++;
        }
    }
    dest[a] = 0;
    return (dest);
}
```

LVL 2 : 2-0-inter

```
Assignment name  : inter
Expected files   : inter.c
Allowed functions: write
--------------------------------------------------------------------------------

Écrire un programme qui prend en paramètres deux chaînes de caractères et qui
affiche sans doublon les caractères communs aux deux chaînes.

L'affichage se fera dans l'ordre d'apparition dans la premiere chaîne.
L'affichage doit être suivi d'un '\n'.

Si le nombre de paramètres transmis est différent de 2, le programme affiche
'\n'.

Exemples:
$>./inter "padinton" "paqefwtdjetyiytjneytjoeyjnejeyj" | cat -e
padinto$
$>./inter ddf6vewg64f gtwthgdwthdwfteewhrtag6h4ffdhsd | cat -e
df6ewg4$
$>./inter "rien" "cette phrase ne cache rien" | cat -e
rien$
$>./inter | cat -e
$
```

```c
#include <unistd.h>

void    ft_putchar(char c)
{
    write (1, &c, 1);
}

int main(int ac, char **av)
{
    int i;
    int j;
    int doub;

    i = 0;
    j = 0;
    if (ac == 3)
    {
        while (av[1][i] && av[2][j])
        {
            while (av[1][i] != av[2][j] && av[2][j] != '\0')
                j++;
            if (av[1][i] == av[2][j])
            {
                doub = i - 1;
                while (av[1][doub] != av[1][i] && doub >= 0)
                    doub--;
                if (av[1][doub] == av[1][i])
                    i++;
                else
                {
                    ft_putchar(av[1][i]);
                    i++;
                }
                j = 0;
            }
            if (av[1][i] == '\0')
            {
                ft_putchar('\n');
                    return (0);
            }
            if (av[2][j] == '\0')
            {
                j = 0;
                i++;
            }
        }
    }
    ft_putchar('\n');
    return (0);
}
```

LVL 2 : 2-0-reverse_bits

```
Assignment name  : reverse_bits
Expected files   : reverse_bits.c
Allowed functions:
--------------------------------------------------------------------------------

Écrire une fonction qui prend un octet, l'inverse bit à bit (comme dans
l'exemple) et retourne le résultat.

Votre fonction doit être déclarée comme suit:

unsigned char   reverse_bits(unsigned char octet);

Exemple:

  1 byte
_____________
 0100  0001
     ||
     \/
 1000  0010
```

```c
#include <stdio.h>

unsigned char reverve_bits(unsigned char octet)
{
   octet = (octet & 0xF0) >> 4 | (octet & 0x0F) << 4;
   octet = (octet & 0xCC) >> 2 | (octet & 0x33) << 2;
   octet = (octet & 0xAA) >> 1 | (octet & 0x55) << 1;
   return octet;
}
int main(void)
{
    unsigned char octet;

    octet = 5;
    printf("%hhu", reverse(octet));
    return (0);
}
```

LVL 2 : 2-0-swap_bits

```
Assignment name  : swap_bits
Expected files   : swap_bits.c
Allowed functions:
--------------------------------------------------------------------------------

Écrire une fonction qui prend un octet, inverse ses moitiés (comme dans l'exemple), puis
retourne le résultat.

Votre fonction doit être déclarée comme suit:

unsigned char   swap_bits(unsigned char octet);

Exemple:

  1 byte
_____________
 0100 | 0001
      \ /
      / \
 0001 | 0100
```

```c
#include <stdio.h>

unsigned char   swap_bits(unsigned char octect)
{
    octect = (((octect & 0xf0) >> 4) | ((octect & 0x0f) << 4));
    return (octect);
}

int main(void)
{
    unsigned char octect;

    octect = 2;
    printf("%hhu", swap_bits(octect));
    return (0);
}
```

LVL 2 :2-0-union

```
Assignment name  : union
Expected files   : union.c
Allowed functions: write
--------------------------------------------------------------------------------

Écrire un programme nommé union qui prend en paramètre deux chaînes de
caractères et qui affiche, sans doublon, les caractères qui apparaissent dans
l'une ou dans l'autre.

L'affichage se fera dans l'ordre d'apparition dans la ligne de commande.

L'affichage doit etre suivi d'un retour à la ligne.

Si le nombre de paramètres transmis est différent de 2, le programme affiche
\n.

Exemple:

$>./union zpadinton "paqefwtdjetyiytjneytjoeyjnejeyj" | cat -e
zpadintoqefwjy$
$>./union ddf6vewg64f gtwthgdwthdwfteewhrtag6h4ffdhsd | cat -e
df6vewg4thras$
$>./union "rien" "cette phrase ne cache rien" | cat -e
rienct phas$
$>./union | cat -e
$
$>
$>./union "rien" | cat -e
$
$>
```

```c
#include <unistd.h>

void    ft_putchar(char c)
{
    write(1, &c, 1);
}

int     ft_strlen(char *str)
{
    int i;

    i = 0;
    while (str[i] != '\0')
        i++;
    return (i);
}

int     check_before(char *str, char c, int place)
{
    int i;

    i = 0;
    while (i <= place)
    {
        if (str[i] == c && i < place)
            return(0);
        i++;
    }
    return(1);
}

void    union_str(char *str1, char *str2)
{
    int i;
    int j;

    i = 0;
    j = 0;
    while (str1[i] != '\0')
    {
        if (check_before(str1, str1[i], i))
            ft_putchar(str1[i]);
        i++;
    }
    while (str2[j] != '\0')
    {
        if (check_before(str1, str2[j], i) && check_before(str2, str2[j], j))
            ft_putchar(str2[j]);
        j++;
    }
}

int     main(int argc, char **argv)
{
    union_str(argv[1], argv[2]);
    return (0);
}
```

LVL 2 : 2-1-alpha_mirror

```
Assignment name  : alpha_mirror
Expected files   : alpha_mirror.c
Allowed functions: write
--------------------------------------------------------------------------------

Écrire un programme alpha_mirror qui prend une chaîne et l'affiche après en avoir remplacé
chaque
caractère alphabétique par le caractère alphabétique opposé, suivie d'un newline.

'a' devient 'z', 'Z' devient 'A'
'd' devient 'w', 'M' devient 'N'

Etc...

La casse n'est pas changée.

Si le nombre d'arguments n'est pas 1, affiche juste un newline.

Exemples:

$>./alpha_mirror "abc"
zyx
$>./alpha_mirror "My horse is Amazing." | cat -e
Nb slihv rh Znzarmt.$
$>./alpha_mirror | cat -e
$
$>
```

```c
#include <unistd.h>

void    ft_putchar(char c)
{
    write(1, &c, 1);
}

int     main(int ac, char **av)
{
    int     i;

    i = 0;
    if (ac == 2)
    {
        while (av[1][i])
        {
            if (av[1][i] >= 'a' && av[1][i] <= 'z')
                av[1][i] = 'z' - (av[1][i] - 'a');
            else if (av[1][i] >= 'A' && av[1][i] <= 'Z')
                av[1][i] = 'Z' - (av[1][i] - 'A');
            ft_putchar(av[1][i]);
            i++;
        }
    }
    ft_putchar('\n');
    return (0);
}
```

LVL 2 : 2-1-max

```
Assignment name  : max
Expected files   : max.c
Allowed functions:
--------------------------------------------------------------------------------

Écrire la fonction suivante:

int     max(int* tab, unsigned int len);

Le premier paramètre est un tableau d'int, le deuxième est le nombre d'éléments
contenus dans ce tableau.

La fonction renvoie le plus grand nombre trouvé dans le tableau.

Si le tableau est vide, la fonction renvoie 0.
```

```c
int     max(int *tab, unsigned int len)
{
    unsigned int i;
    int max;

    i = 0;
    max = -214748348;
    if (tab != NULL)
    {
        while (i < len)
        {
            if (tab[i] > max)
                max = tab[i];
            i++;
        }
    return (max);
    }
    else
        return (0);
}
```

LVL 2 : wdmatch

```
Assignment name  : wdmatch
Expected files   : wdmatch.c
Allowed functions: write
--------------------------------------------------------------------------------

Le programme prend en paramètres deux chaînes de caractères et vérifie qu'il
est possible d'écrire la première chaîne de caractères a l'aide des caractères
de la deuxième chaîne, tout en respectant l'ordre des caractères dans la
deuxième chaîne.

Si cela est possible, le programme affiche la première chaîne de caractères,
suivie de '\n'. Sinon le programme affiche seulement '\n'.

Si le nombre de paramètres transmis est différent de 2, le programme affiche
'\n'.

Exemples :

$>./wdmatch "faya" "fgvvfdxcacpolhyghbreda" | cat -e
faya$
$>./wdmatch "faya" "fgvvfdxcacpolhyghbred" | cat -e
$
$>./wdmatch "quarante deux" "qfqfsudf arzgsayns tsregfdgs sjytdekuoixq " | cat -e
quarante deux$
$>./wdmatch "error" rrerrrfiiljdfxjyuifrrvcoojh | cat -e
$
$>./wdmatch | cat -e
$
```

```c
#include <unistd.h>

void    ft_putchar(char c)
{
    write(1, &c, 1);
}

int     main(int ac, char **av)
{
    int     i;
    int     j;

    i = 0;
    j = 0;
    if (ac == 3)
    {
        while (av[1][i] && av[2][j])
        {
            if (av[1][i] == av[2][j])
                i++;
            j++;
        }
        if (av[1][i] == 0)
        {
            i = 0;
            while (av[1][i])
            {
                ft_putchar(av[1][i]);
                i++;
            }
        }
    }
    ft_putchar('\n');
    return (0);
}
```

LVL 2 : do_op

```
Assignment name  : do_op
Expected files   : *.c, *.h
Allowed functions: atoi, printf, write
--------------------------------------------------------------------------------

Écrire un programme qui prend en paramètre trois chaînes de caractères:
- La première et la troisième sont des représentations en base 10 de
  nombre entiers signés, tenant dans des int.
- La deuxième est un opérateur arithmétique parmi $+ - * / \%$.

Le programme doit afficher le resultat de l'opération arithmétique demandée,
suivi d'un '\n'. Si le nombre de paramètres est différent de 3, le programme
doit afficher '\n'.

Vous pouvez partir du principe que les chaînes ne comportent ni erreur, ni
caractères indésirables. Les nombres négatifs, en entrée comme en sortie,
seront précédés d'un et un seul signe '-'. Le résultat de l'opération tient
dans un int.

Exemples:

$> ./do_op "123" "*" 456 | cat -e
56088$
$> ./do_op "9828" "/" 234 | cat -e
42$
$> ./do_op "1" "+" "-43" | cat -e
-42$
$> ./do_op | cat -e
$
```

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

int     calc(int a, int b, char c)
{
    if (c == '+')
        return (a + b);
    else if (c == '-')
        return (a - b);
    else if (c == '*')
        return (a * b);
    else if (c == '/')
        return (a / b);
    return (a % b);
}

int     main(int ac, char **av)
{
    int     res;

    if (ac == 4)
    {
        res = calc(atoi(av[1]), atoi(av[3]), av[2][0]);
        printf("%d", res);
    }
    printf("\n");
}
```

LVL 2 : ft_print_bits

```
Assignment name  : print_bits
Expected files   : print_bits.c
Allowed functions: write
--------------------------------------------------------------------------------

Écrire une fonction qui prend un octet et l'affiche en binaire, sans newline à la fin.

Votre fonction doit être déclarée comme suit:

void    print_bits(unsigned char octet);

Exemple, si on lui passe 2, elle affiche "00000010"
```

```c
#include <unistd.h>

void    ft_putchar(char c)
{
    write(1, &c, 1);
}

void    print_bits(unsigned char octet)
{
    int puissance;

    puissance = 128;
    while (puissance > 0)
    {
        if (octet >= puissance)
        {
            ft_putchar('1');
            octet -= puissance;
            puissance /= 2;
        }
        else
        {
            ft_putchar('0');
            puissance /= 2;
        }
    }
}
```

LVL 2 : ft_strrev

```
Assignment name  : ft_strrev
Expected files   : ft_strrev.c
Allowed functions:
--------------------------------------------------------------------------------

Écrire une fonction qui inverse (en place) une chaîne de caractères.

Elle devra renvoyer son paramètre.

Votre fonction devra être prototypée de la façon suivante :

char    *ft_strrev(char *str);
```

```c
char    *ft_strrev(char *str)
{
    char    temp;
    int     i;
    int     j;

    i = 0;
    j = 0;
    while (str[i] != '\0')
    {
        i++;
    }
    i--;
    while (j <= i)
    {
        temp = str[j];
        str[j] = str[i];
        str[i] = temp;
        i--;
        j++;
    }
    return (str);
}
```

LVL 2 : ft_strcmp

```
Assignment name  : ft_strcmp
Expected files   : ft_strcmp.c
Allowed functions:
--------------------------------------------------------------------------------

Reproduisez à l'identique le comportement de la fonction strcmp (man strcmp).

Votre fonction devra être prototypée de la façon suivante :

int    ft_strcmp(char *s1, char *s2);
```

```c
#include <stdio.h>
#include <string.h>
int     ft_strcmp(const char *s1, const char *s2)
{
    int i;

    i = 0;
    while (s1[i] == s2[i])
    {
        if (s1[i] == '\0' && s2[i] == '\0')
            return (0);
        i++;
    }
    return (s1[i] - s2[i]);
}

int main(int ac, char **av)
{
    char s1[6] = "badqo";
    char s2[6] = "baboo";
    printf("%d   %d",ft_strcmp(s1, s2),strcmp(s1, s2));
    return(0);
}
```

LVL 2 : last_word

```
Assignment name  : last_word
Expected files   : last_word.c
Allowed functions: write
--------------------------------------------------------------------------------

Écrire un programme qui prend une chaîne de caractères en paramètre, et qui
affiche le dernier mot de cette chaîne, suivi d'un '\n'.

On appelle "mot" une portion de chaîne de caractères délimitée soit par des
espaces et/ou des tabulations, soit par le début / fin de la chaîne.

Si le nombre de paramètres transmis est différent de 1, ou s'il n'y a aucun mot
à afficher, le programme affiche '\n'.

Exemple:

$> ./last_word "FOR PONY" | cat -e
PONY$
$> ./last_word "this        ...       is sparta, then again, maybe    not" | cat -e
not$
$> ./last_word "   " | cat -e
$
$> ./last_word "a" "b" | cat -e
$
$> ./last_word "  lorem,ipsum  " | cat -e
lorem,ipsum$
$>
```

```c
#include <unistd.h>

int main(int argc, char **argv)
{
    int i;
    int has_word = 0;

    i = 0;
    if (argc != 2 || argv[1][0] == '\0')
    {
        write(1, "\n", 1);
        return 0;
    }
    while (argv[1][i] != '\0')
    {
        if (argv[1][i] > 32)
            has_word = 1;
        i++;
    }
    if (has_word == 0)
    {
        write(1, "\n", 1);
        return 0;
    }
    while (argv[1][i] < 33)
        i--;
    while (argv[1][i] > 32)
        i--;
    i++;
    while (argv[1][i] > 32)
    {
        write(1, &argv[1][i], 1);
        i++;
    }
    write(1, "\n", 1);
    return 0;
}
```

LVL 3 : 3-0-ft_list_size

```
Assignment name  : ft_list_size
Expected files   : ft_list_size.c, ft_list.h
Allowed functions:
--------------------------------------------------------------------------------

Écrire une fonction qui renvoie le nombre d'éléments dans la liste chaînée
passée en paramètre.

Elle devra être prototypée de la façon suivante:

int ft_list_size(t_list *begin_list);

Vous devez utiliser la structure suivante, et la rendre dans un fichier
ft_list.h:

typedef struct    s_list
{
    struct s_list *next;
    void          *data;
}                 t_list;
```

Fonction :

```c
#include "ft.h"

int     ft_list_size(t_list *begin_list)
{
    int i;

    i = 0;
    while (begin_list->next)
    {
        begin_list = begin_list->next;
        i++;
    }
    return (i + 1);
}
```

header :

```c
#ifndef     FT_H
# define    FT_H

typedef struct    s_list
{
    struct s_list *next;
    void          *data;
}                 t_list;
#endif
```

LVL 3 : 3-0-hidenp

```
Assignment name  : hidenp
Expected files   : hidenp.c
Allowed functions: write
--------------------------------------------------------------------------------

Écrire un programme nommé hidenp qui prend en paramètres deux chaînes de
caractères et qui affiche 1 suivi de '\n' si la première chaîne est cachée dans
la deuxième chaîne. Sinon il affiche 0 suivi de '\n'.

Soit s1 et s2 des chaînes de caractères. On dit que la chaîne s1 est cachée
dans la chaîne s2 si on peut trouver chaque caractère de s1 dans s2 et ce dans
le même ordre que dans s1. En outre, la chaîne vide est cachée dans n'importe
quelle chaîne.

Si le nombre de paramètres transmis est différent de 2, le programme affiche
'\n'.

Exemples :

$>./hidenp "fgex.;" "tyf34gdgf;'ektufjhgdgex.;.;rtjynur6" | cat -e
1$
$>./hidenp "abc" "2altrb53c.sse" | cat -e
1$
$>./hidenp "abc" "btarc" | cat -e
0$
$>./hidenp | cat -e
$
$>
```

```c
#include <unistd.h>

void    ft_putchar(char c)
{
    write(1, &c, 1);
}

int     main(int ac, char **av)
{
    int     i;
    int     j;

    i = 0;
    j = 0;
    if (ac == 3)
    {
        while (av[1][i] && av[2][j])
        {
            if (av[1][i] == av[2][j])
                i++;
            j++;
        }
        if (av[1][i])
            ft_putchar('0');
        else
            ft_putchar('1');
    }
    ft_putchar('\n');
    return (0);
}
```

LVL 3 : 3-0-PGCD

```
Assignment name  : pgcd
Expected files   : pgcd.c
Allowed functions: printf, atoi, malloc, free
--------------------------------------------------------------------------------

Écrire un programme qui prend deux chaînes de caractères représentant des
nombres entiers positifs non nuls en paramètre.

Les entiers représentés par les paramètres tiennent dans un int.

Afficher le plus grand diviseur commun à ces deux nombres suivi de '\n'. Le
PGCD est toujours un entier positif non nul.

Si le nombre de paramètres est différent de 2, le programme affiche seulement
'\n'.

Exemple:

$> ./pgcd 42 10 | cat -e
2$
$> ./pgcd 42 12 | cat -e
6$
$> ./pgcd 14 77 | cat -e
7$
$> ./pgcd 17 3 | cat -e
1$
$> ./pgcd | cat -e
$
```

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

int     main(int ac, char **av)
{
    int     pgcd;
    int     a;
    int     b;

    if (ac == 3)
    {
        a = atoi(av[1]);
        b = atoi(av[2]);
        pgcd = a % b;
        while (pgcd != 0)
        {
            a = b;
            b = pgcd;
            pgcd = a % b;
        }
        printf("%d", b);
    }
    printf("\n");
    return (0);
}
```

LVL 3 : 3-0-print_hex

```
Assignment name  : print_hex
Expected files   : print_hex.c
Allowed functions: write
--------------------------------------------------------------------------------

Écrire un programme qui prend un entier positif ou nul exprimé en base 10, et l'affiche en base 16 (avec des
minuscules) suivi d'un newline.

Si le nombre de paramêtres n'est pas 1, le programme affiche un newline.

Exemples:

$> ./print_hex "10" | cat -e
a$
$> ./print_hex "255" | cat -e
ff$
$> ./print_hex "5156454" | cat -e
4eae66$
$> ./print_hex | cat -e
$
```

```c
#include <unistd.h>

void    ft_putchar(char c)
{
    write(1, &c, 1);
}

void    ft_print_hex(long a)
{
    if (a >= 16)
    {
        ft_print_hex(a / 16);
        ft_print_hex(a % 16);
    }
    else if (a < 10)
        ft_putchar(a + 48);
    else
        ft_putchar('a' + a - 10);

}
int     main(int argv, char **argc)
{
    int i;
    long a;

    i = 0;
    a = 0;
    if (argv == 2)
    {
        while (argc[1][i] != '\0')
        {
            if (!(argc[1][i] <= '9' && argc[1][i] >= '0'))
            {
                ft_putchar('\n');
                return (0);
            }
            a = a * 10 + (argc[1][i] - '0');
            i++;
        }
        ft_print_hex(a);
    }
    ft_putchar('\n');
    return (0);
}
```

LVL 3 : 3-0-ft _range

```
Assignment name  : ft_range
Expected files   : ft_range.c
Allowed functions: malloc
--------------------------------------------------------------------------------

Écrire la fonction suivante :

int     *ft_range(int start, int end);

Cette fonction doit allouer avec malloc() un tableau d'ints, le remplir avec
les valeurs (consécutives) démarrant à start et finissant à end (start et end
inclus !), et renvoyer un pointeur vers la première valeur du tableau.

Exemples:

- Avec (1, 3) vous devrez renvoyer un tableau contenant 1, 2 et 3.
- Avec (-1, 2) vous devrez renvoyer un tableau contenant -1, 0, 1 et 2.
- Avec (0, 0) vous devrez renvoyer un tableau contenant 0.
- Avec (0, -3) vous devrez renvoyer un tableau contenant 0, -1, -2 et -3.
```

```c
#include <stdlib.h>

int     *ft_rrange(int start, int end)
{
    int *tab;
    int range;

    range = start - end;
    if (range < 0)
        range *= -1;
    tab = (int *)malloc(sizeof(*tab) * range);
    while (range >= 0)
    {
        if (end > start)
            tab[range] = end - range;
        if (end < start)
            tab[range] = end + range;
        range--;
    }
    return (tab);
}
```

LVL 3 : 3-0-epur_str

```
Assignment name  : epur_str
Expected files   : epur_str.c
Allowed functions: write
--------------------------------------------------------------------------------

Écrire un programme qui prend une chaîne de caractères en paramètre, et qui
affiche cette chaîne avec exactement un espace entre chaque mot, sans espaces
ou tabulations ni au début ni à la fin de la chaîne, suivie d'un '\n'.

On appelle "mot" une portion de chaîne de caractères délimitée soit par des
espaces et/ou des tabulations, soit par le début / fin de la chaîne.

Si le nombre de paramètres transmis est différent de 1, ou s'il n'y a aucun mot
à afficher, le programme affiche '\n'.

Exemple:

$> ./epur_str "vous voyez c'est facile d'afficher la meme chose" | cat -e
vous voyez c'est facile d'afficher la meme chose$
$> ./epur_str " seulement          la c'est      plus dur " | cat -e
seulement la c'est plus dur$
$> ./epur_str "comme c'est cocasse" "vous avez entendu, Mathilde ?" | cat -e
$
$> ./epur_str "" | cat -e
$
$>
```

```c
#include <unistd.h>

int     main(int argc, char **argv)
{
    int i;
    int has_word;

    i = 0;
    has_word = 0;
    if (argc != 2 || argv[1][0] == '\0')
    {
        write (1, "\n", 1);
        return 0;
    }
    while (argv[1][i] != '\0')
    {
        while (argv[1][i] < 33)
        {
            if (argv[1][i] == '\0')
            {
                write (1, "\n", 1);
                return 0;
            }
            i++;
        }
        if (has_word != 0)
            write (1, " ", 1);
        while (argv[1][i] > 32)
        {
            write (1, &argv[1][i], 1);
            i++;
        }
        has_word = 1;
    }
    write (1, "\n", 1);
    return 0;
}
```

LVL 3 : 3-1-expand_str

```
Assignment name  : expand_str
Expected files   : expand_str.c
Allowed functions: write
--------------------------------------------------------------------------------

Écrire un programme qui prend une chaîne de caractères en paramètre, et qui
affiche cette chaîne avec exactement trois espaces entre chaque mot, sans
espaces ou tabulations ni au début ni à la fin de la chaîne, suivie d'un '\n'.

On appelle "mot" une portion de chaîne de caractères délimitée soit par des
espaces et/ou des tabulations, soit par le début / fin de la chaîne.

Si le nombre de paramètres transmis est différent de 1, ou s'il n'y a aucun mot
à afficher, le programme affiche '\n'.

Exemple:

$> ./expand_str "vous   voyez   c'est   facile   d'afficher   la   meme   chose" | cat -e
vous   voyez   c'est   facile   d'afficher   la   meme   chose$
$> ./expand_str " seulement          la c'est      plus dur " | cat -e
seulement   la   c'est   plus   dur$
$> ./expand_str "comme c'est cocasse" "vous avez entendu, Mathilde ?" | cat -e
$
$> ./expand_str "" | cat -e
$
$>
```

```c
#include <unistd.h>

void    ft_putchar(char c)
{
    write(1, &c, 1);
}

int     main(int ac, char **av)
{
    int     i;
    int     j;

    i = 0;
    if (ac == 2)
    {
        while (av[1][i] == ' ' && av[1][i])
            i++;
        while (av[1][i])
        {
            while (av[1][i] != ' ' && av[1][i])
            {
                ft_putchar(av[1][i]);
                i++;
            }
            j = i;
            while (av[1][j] == ' ' && av[1][j])
                j++;
            if (av[1][j])
                write(1, "   ", 3);
            i = j;
        }
    }
    ft_putchar('\n');
    return (0);
}
```

LVL 3 : 3-2-atoi_base

```
Assignment name  : ft_atoi_base
Expected files   : ft_atoi_base.c
Allowed functions: None
--------------------------------------------------------------------------------

Écrire une fonction convertit son argument 'str', une chaîne (en base N <= 16)
en un entier (base 10) et le retourne.

Les caractères reconnus dans l'entrée sont : 0123456789abcdef Bien entendu, la
base demandée conditionne le nombre de caractères à prendre en compte. Par
exemple, la base 4 reconnaîtra "0123" et la base 16 reconnaîtra
"0123456789abcdef".

Les majuscules marchent aussi : "12fdb3" est pareil que "12FDB3".

Les caractères '-' doivent être interprétés seulement s'ils sont en première
position dans la chaîne.

Votre fonction sera déclarée comme suit:

int ft_atoi_base(const char *str, int str_base);
```

```c
int     ft_atoi_base(const char *str, int str_base)
{
    int     i;
    int     res;
    int     sign;

    sign = 1;
    i = 0;
    while (str[i] == ' ' || str[i] == '\t')
        i++;
    if (str[1] == '+' || str[1] == '-')
    {
        if (str[1] == '-')
            sign = -1;
        i++;
    }
    while ((str[i]) && ((str[i] >= '0' && str[i] <= '9') || (str_base > 10 &&
                str[i] - 'a' <= (str_base % 10) && str[i] - 'a' >= 0) ||
            (str_base > 10 && str[i] - 'A' <= (str_base % 10) &&
             str[i] - 'A' >= 0)))
    {
        if (str[i] >= 'a' && str[i] - 'a' <= str_base % 10)
            res = res * str_base + (10 + str[i] - 'a');
        else if (str[i] >= 'A' && str[i] - 'A' <= str_base % 10)
            res = res * str_base + (10 + str[i] - 'A');
        else
            res = res * str_base + (str[i] - '0');
        i++;
    }
    return (res * sign);
}
```

LVL 3 : add_prime_sum

```
Assignment name  : add_prime_sum
Expected files   : add_prime_sum.c
Allowed functions: write, exit
--------------------------------------------------------------------------------

Écrire un programme qui prend un entier positif en argument et affiche la somme
de tous les nombres premiers inférieurs ou égaux à ce paramètre, suivie d'un
\n.

Si le nombre d'arguments n'est pas 1, ou que l'argument n'est pas positif,
afficher 0 et un \n.

Oui, les exemples sont justes.

Exemples:

$>./add_prime_sum 5
10
$>./add_prime_sum 7 | cat -e
17$
$>./add_prime_sum | cat -e
0$
$>
```

```c
#include <stdio.h>
#include <unistd.h>

void    ft_putchar(char c)
{
    write(1, &c, 1);
}

void   ft_putnbr(int nb)
{
    if (nb >= 10)
        ft_putnbr(nb / 10);
    ft_putchar(nb % 10 + '0');
}

int  ft_atoi(char *str)
{
    int negatif;
    int i;
    int res;

    res = 0;
    i = 0;
    while (str[i] == '\t' || str[i] == '\v' || str[i] == '\r' || str[i] == '\f' || str[i] == '
        i++;
    if (str[i] == '-' || str[i] == '+')
    {
        if (str[i] == '-')
        {
            negatif = 1;
        }
        i++;
    }
    while (str[i] >= '0' && str[i] <= '9')
    {
        res = res * 10 + (str[i] - '0');
        i++;
    }
    if (negatif == 1)
        res = -res;
    return (res);
}
```

I I

Suite du code

I I

V

```c
int  is_prime(unsigned int nb)
{
    unsigned int i;

    i = 2;
    if (nb == 2)
        return (1);
    if (!(nb % 2) || nb == 1)
        return (0);
    while (i < nb)
    {
        if (!(nb % i))
            return (0);
        i++;
    }
    return (1);
}

unsigned int  add_prime(unsigned int nb)
{
    int res;

    res = 0;
    while (nb > 1)
    {
        if (is_prime(nb))
            res = res + nb;
        nb--;
    }
    return (res);
}

int    main(int argc, char **argv)
{
    int nb;

    nb = ft_atoi("220000");
    if (nb <= 0)
    {
        write(1, "0\n", 2);
        return (0);
    }
    ft_putnbr(add_prime(nb));
    return (0);
}
```

LVL 3 : putnbr_base

```c
#include <unistd.h>
#include <stdlib.h>

void    ft_putchar(char c)
{
  write(1, &c, 1);
}

int ft_strlen(char *str)
{
  int i;

  i = 0;
    while (str[i])
      ++i;
    return (i);
}

void    ft_putnbr(int nb, char *base)
{
  int   len_base;

  len_base = ft_strlen(base);
    if (nb == -2147483648)
    {
      ft_putnbr(nb / len_base, base);
      ft_putchar('8');
    }
    if (nb >= -2147483647 && nb < 0)
    {
        ft_putchar('-');
        ft_putnbr(-nb, base);
    }
    else if (nb <= 2147483647 && nb >= len_base)
    {
      ft_putnbr(nb / len_base, base);
      ft_putnbr(nb % len_base, base);
    }
    else if (nb <= len_base)
    {
        if (nb >= 0)
            ft_putchar(base[nb]);
    }
}

int main(int ac, char **av)
{
  (void)ac;

  ft_putnbr(atoi(av[1]), av[2]);
  return (0);
}
```

LVL 4 : 4-0-fprime

```
Assignment name  : fprime
Expected files   : fprime.c
Allowed functions: printf, atoi
--------------------------------------------------------------------------------

Écrire un programme qui prend en paramètre un entier strictement positif, et
qui affiche sa décomposition en facteurs premiers sur la sortie standard,
suivie d'un '\n'.

Les facteurs doivent être affichés dans l'ordre croissant et séparés par des
'*', de telle sorte que l'expression affichée donne le bon résultat.

Si le nombre de paramètres est différent de 1, le programme doit afficher '\n'.

L'entrée, quand elle est passée, sera toujours un nombre valide sans caractères
parasites.

Exemple:

$> ./fprime 225225 | cat -e
3*3*5*5*7*11*13$
$> ./fprime 8333325 | cat -e
3*3*5*5*7*11*13*37$
$> ./fprime 9539 | cat -e
9539$
$> ./fprime 804577 | cat -e
804577$
$> ./fprime 42 | cat -e
2*3*7$
$> ./fprime 1 | cat -e
1$
$> ./fprime | cat -e
$
$> ./fprime 42 21 | cat -e
$
```

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

/*
Sans doute moyen de faire plus propre mais ca fonctionne
apres ca segfault sur zero mais je sais pas trop si ya un test sur le zero
*/

unsigned int    g_flist[64];
int             g_i;

void    fprime(unsigned int n, int d)
{
    if (n != 1)
    {
        while (n % d)
            d++;
        g_flist[g_i++] = d;
        fprime(n / d, d);
    }
}

int     main(int argc, char **argv)
{
    unsigned int    n;
    int             i;

    g_i = 0;
    i = 0;
    if (argc - 1 == 1)
    {
        n = atoi(argv[argc - 1]);
        if (n == 1)
        {
            printf("1\n");
            return (0);
        }
        fprime(n, 2);

        while (i < g_i)
        {
            printf("%u", g_flist[i]);
            if (i < g_i - 1)
                printf("*");
            i++;
        }
    }
    printf("\n");
    return (0);
}
```

LVL 4 : 4-4-ft_itoa

```
Assignment name  : ft_itoa
Expected files   : ft_itoa.c
Allowed functions: malloc
--------------------------------------------------------------------------------

Écrire une fonction qui prend un int et le convertit en chaîne terminée par un
caractère nul. Cette fonction retourne le résultat en tant qu'un tableau de
char que vous devez allouer.

Votre fonction sera déclarée comme suit:

char    *ft_itoa(int nbr);
```

```c
#include <stdlib.h>
#include "libft.h"

static  int         count(int n)
{
    int a;

    a = 1;
    while (n /= 10)
        a++;
    return (a);
}

static char         *files_max_int(int n)
{
    char *str;

    str = malloc(sizeof(char) * 11);
    str = ft_itoa(-(n - 1));
    str[10]++;
    return (str);
}

char                *ft_itoa(int n)
{
    char    *str;
    int     a;

    a = count(n) + (n < 0) - 1;
    str = malloc(count(n) + (n < 0) + 1);
    str[count(n) + (n < 0) + 1] = 0;
    if (n == -2147483648)
        return (files_max_int(n));
    if (!n)
        str[0] = '0';
    if (n < 0)
    {
        n *= -1;
        str[0] = '-';
    }
    str[a + 1] = 0;
    while (n)
    {
        str[a--] = n % 10 + 48;
        n /= 10;
    }
    return (str);
}
```