B


```c
#include "files_entier.h"

/* QUESTION 3 :

La file vide est reprÃ©sentÃ©e par ififo_s avec comme pointeur NULL suivant.

*/

/* QUESTION 4 : */

ififo_s *ififo_new(){

    ififo_node_s *newnoeud = NULL;
    ififo_s *files = malloc(sizeof(*files));

    if(files == NULL){
        return NULL;
    }

    files -> suivant = newnoeud;
    files -> dernier = newnoeud;

    return files;
}

/* QUESTION 5 : */

int ififo_is_empty(ififo_s *f){

    return ((f -> suivant == NULL) && (f -> dernier == NULL));
}

/* QUESTION 6 : */

int ififo_enqueue(ififo_s *f, int nb){

    ififo_node_s *new = malloc(sizeof(*new));

    if (f == NULL || new == NULL){
        return -1;
    }

    new->nombre = nb;

    if (ififo_is_empty(f)){
        f -> suivant = new;
        f -> dernier = new;
    }
    else{
        f -> dernier -> noeud = new;
        f -> dernier = new;
    }

    new -> noeud = NULL;
    return 0;
}

/* QUESTION 7 : */

int ififo_dequeue(ififo_s *f, int *nb){

    if (ififo_is_empty(f)){
        return -1;
    }

    ififo_node_s *new = f -> suivant;

    if (f -> suivant == f -> dernier){
        ififo_node_s *new2 = NULL;
```



```c
        *nb = f -> suivant -> nombre;
        f -> suivant = f -> dernier = new2;
    }
    else {
        *nb = new -> nombre;
        f -> suivant = new -> noeud;
    }

    free(new);

    return 0;
}

/* QUESTION 8 : */

int ififo_head(const struct ififo_s *f){

    return f -> suivant -> nombre;
}

/* QUESTION 9 : */

int ififo_apply(ififo_s *f, func_t *fn){

    ififo_node_s *apply = f -> dernier;
```

"suivant"

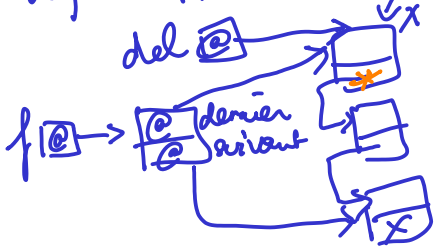
```c
    while (apply != NULL){
        fn(apply -> nombre);
        apply = apply -> noeud;
    }

    return 0;
}

/* QUESTION 10 : */

void ififo_del(ififo_s *f){

    ififo_node_s *new = f -> suivant;
    ififo_node_s *del;


    while(new != NULL){
        del = new;
        new = new -> noeud;
        free(del);
    }
    new = NULL;
    free(f);
}

void print_int(int i){
    printf("%d âM-^FM-^P ", i);
}
```

attention à utiliser des noms qui ont du sens et facilitent la lisibilité du code.

inutile : variable locale.

```c
#include "files_generique.h"

#define X 5 )?

/* QUESTION 1 : (suite) */

gfifo_s *gfifo_new(){

    gfifo_node_s *newnoeud = NULL; inutile
    gfifo_s *files = malloc(sizeof(*files));

    if (files == NULL){
        return NULL;
    }
    files -> suivant = newnoeud;
    files -> dernier = newnoeud;  -> NULL

    return files;
}

int gfifo_del(struct gfifo_s *f){

    gfifo_node_s *del = f -> dernier;

    while (del != NULL){
        free(del);
        f -> dernier = f -> dernier -> noeud;
        del = f->dernier;
    }
    free(f);
    return 0;
}

int gfifo_size(struct gfifo_s *f){

    gfifo_node_s *size = f -> dernier;

    int cpt;
    cpt = 0;

    while (size != NULL){
        cpt++;
        size = size -> noeud;
    }

    return cpt;
}

int gfifo_enqueue(struct gfifo_s *f, void *nb){

    gfifo_node_s *enqueue = malloc(sizeof(*enqueue));

    if (f == NULL || enqueue == NULL){
        return -1;
    }
    -> si f==NULL, faire free(enqueue)

    enqueue -> nombre = nb;
    enqueue -> noeud = NULL;

    if (f -> suivant == NULL && f -> dernier == NULL){
        f -> dernier = f -> suivant = enqueue;
    }
    else{
        f -> suivant -> noeud = enqueue;
        f -> suivant = enqueue;
    }

    return 0;
}
```

trop tard : le noeud est déjà supprimé : free(del)




```c
int gfifo_dequeue(struct gfifo_s *f, void **nb){

    gfifo_node_s *dequeue;

    if (f -> suivant == NULL && f -> dernier == NULL){
        return -1;
    }

    dequeue = f -> dernier;
    (*nb) = dequeue -> nombre;
    f -> dernier = dequeue -> noeud;

    free(dequeue);

    return 0;
}

int gfifo_apply(struct gfifo_s *f, gfunc_t *fn){

    gfifo_node_s *apply = f -> dernier;

    while (apply != NULL){
        fn(apply -> nombre);
        apply = apply -> noeud;
    }

    return 0;
}

void printg_int(void *i){
    printf("âM-^FM-^R %d ", *((int *)i));
}
```

+ cas particulier : suppression du dernier élément

Votre file est "à l'envers" : dernier est en fait la tête de file.

Deux versions du même exercice; lequel dois-je corriger?


```c
#include "gfifo.h"

gfifo_s *gfifo_new(){

    gfifo_s *f = malloc(sizeof(*f));
    if (f == NULL)
        return NULL;

    f -> suivant = NULL;
    f -> dernier = NULL;

    return f;
}

int gfifo_del(gfifo_s *f){

    gfifo_node_s *del = f -> suivant;

    while (del != NULL){
        f -> suivant = f -> suivant -> noeud;
        free(del);
        del = f -> suivant;
    }
    free(f);
    return 0;
}

int gfifo_size(gfifo_s *f){

    gfifo_node_s *size = f -> suivant;

    int cpt = 0;

    while (size != NULL){
        cpt ++;
        size = size -> noeud;
    }
    return cpt;
}

int gfifo_enqueue(gfifo_s *f, void *nb){

    gfifo_node_s *enqueue = malloc(sizeof(*enqueue));

    if (enqueue == NULL)
        return -1;

    enqueue -> nombre = nb;
    enqueue -> noeud = NULL;

    if (f -> suivant == NULL && f -> dernier == NULL){
        f -> suivant = enqueue;
    }
    else{
        f -> dernier -> noeud = enqueue;
    }

    f -> dernier = enqueue;
    return 0;
}

int gfifo_dequeue(gfifo_s *f, void **nb){

    if (f -> suivant == NULL && f -> dernier == NULL){
        return -1;
    }

    gfifo_node_s *suivant = f -> suivant;
    *nb = f -> suivant -> nombre;
```

dk ici.


```c
    if (f -> suivant -> noeud == NULL){
        f -> suivant = NULL;
        f -> dernier = NULL;
    }
    else{
        f -> suivant = f -> suivant -> noeud;
    }
    free(suivant);
    return 0;
}

int gfifo_apply(gfifo_s *f, gfunc_t *fn){
    gfifo_node_s *func = f -> suivant;

    while (func != NULL){
        fn((func -> nombre));
        func = to_func->noeud;
    }
    return 0;
}

void print_int(int i){
    printf("%d âM-^FM-^P ", i);
}

void test_gifo_int(){
    struct gfifo_s *fifo;
    int i;

    fifo = gfifo_new();

    gfifo_enqueue(fifo, 12); /* âM-^FM-^R 12 âM-^FM-^R */
    gfifo_enqueue(fifo, 13); /* âM-^FM-^R 13 âM-^FM-^R 12 âM-^FM-^R */

    gfifo_apply(fifo, print_int);
    putchar('\n');
    printf("size : %d\n", gfifo_size(fifo));
    gfifo_enqueue(fifo, 14); /* âM-^FM-^R 14 âM-^FM-^R 13 âM-^FM-^R 12 âM-^FM-^R
 */
    gfifo_dequeue(fifo, &i); /* 12 & âM-^FM-^R 14 âM-^FM-^R 13 âM-^FM-^R */

    gfifo_apply(fifo, print_int);
    putchar('\n');

    gfifo_dequeue(fifo, &i); /* 13 & âM-^FM-^R 14 âM-^FM-^R */
    gfifo_dequeue(fifo, &i); /* 14 & âM-^FM-^R âM-^FM-^R */
    gfifo_apply(fifo, print_int);
    putchar('\n');

    gfifo_del(fifo);
}

/*
int main(){
    test_gifo_int();
    return 0;
}
*/
```

pas pour les files génériques.

Idem


```c
#include "ififo.h"
#include "gfifo.h"

ififo_s *ififo_new(){

    struct ififo_s *f = malloc(sizeof(*f));
    if (f == NULL)
        return NULL;

    f -> suivant = NULL;
    f -> dernier = NULL;

    return f;
}

int ififo_is_empty(ififo_s *f){
    return (f -> suivant == NULL && f -> dernier == NULL);
}

int ififo_enqueue(ififo_s *f, int nb){

    ififo_node_s *new = malloc(sizeof(*new));

    if (new == NULL){
        return -1;
    }

    new -> nombre = nb;
    new -> noeud = NULL;

    if (ififo_is_empty(f)){
        f -> suivant = new;
    }
    else{
        f -> dernier -> noeud = new;
    }

    f -> dernier = new;
    return 0;
}

int ififo_dequeue(ififo_s *f, int *nb){

    if (ififo_is_empty(f)){
        return -1;
    }

    ififo_node_s *suivant = f -> suivant;
    *nb = suivant -> nombre;

    f -> suivant = suivant -> noeud;
    free(suivant);
    return 0;
}

int ififo_head(const struct ififo_s *f){
    return f -> suivant -> nombre;
}

int ififo_apply(ififo_s *f, func_t *fn){
    ififo_node_s *func = f -> suivant;

    while (func != NULL){
        fn(func -> nombre);
        func = func -> noeud;
    }
    return 0;
}
```

Idem ici.


```c
void ififo_del(struct ififo_s *f){

    ififo_node_s *del = f -> suivant;

    while (del != NULL){
        f -> suivant = f -> suivant -> noeud;
        free(del);
        del = f -> suivant;
    }
    free(f);
}

void test_fifo_int(){
    struct ififo_s *fifo;
    int i;

    fifo = ififo_new();

    ififo_enqueue(fifo, 12); /* âM-^FM-^R 12 âM-^FM-^R */
    ififo_enqueue(fifo, 13); /* âM-^FM-^R 13 âM-^FM-^R 12 âM-^FM-^R */

    ififo_apply(fifo, print_int);
    putchar('\n');

    ififo_enqueue(fifo, 14); /* âM-^FM-^R 14 âM-^FM-^R 13 âM-^FM-^R 12 âM-^FM-^R
 */
    ififo_dequeue(fifo, &i); /* 12 & âM-^FM-^R 14 âM-^FM-^R 13 âM-^FM-^R */

    printf("%d\n", i);
    ififo_apply(fifo, print_int);
    putchar('\n');

    ififo_dequeue(fifo, &i); /* 13 & âM-^FM-^R 14 âM-^FM-^R */
    ififo_dequeue(fifo, &i); /* 14 & âM-^FM-^R âM-^FM-^R */
    ififo_apply(fifo, print_int);
    putchar('\n');

    ififo_del(fifo);
}
/*
int main(){
    test_fifo_int();
    return 0;
}
*/
```

```c
#include "files_entier.h"

/* make main_files_entier */

void test_fifo_int()
{
    struct ififo_s *fifo;
    int i;

    fifo = ififo_new();

    ififo_enqueue(fifo, 12); /* âM-^FM-^R 12 âM-^FM-^R */
    ififo_enqueue(fifo, 13); /* âM-^FM-^R 13 âM-^FM-^R 12 âM-^FM-^R */

    ififo_apply(fifo, print_int);
    putchar('\n');

    ififo_enqueue(fifo, 14); /* âM-^FM-^R 14 âM-^FM-^R 13 âM-^FM-^R 12 âM-^FM-^R
 */
    ififo_dequeue(fifo, &i); /* 12 & âM-^FM-^R 14 âM-^FM-^R 13 âM-^FM-^R */

    printf("%d\n", i);
    ififo_apply(fifo, print_int);
    putchar('\n');

    ififo_dequeue(fifo, &i); /* 13 & âM-^FM-^R 14 âM-^FM-^R */
    ififo_dequeue(fifo, &i); /* 14 & âM-^FM-^R âM-^FM-^R */
    ififo_apply(fifo, print_int);
    putchar('\n');

    ififo_del(fifo);
}

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

```c
#include "files_generique.h"
#define X 5

/* make main_files_generique*/

/* QUESTION 2 : */

void test_gfifo(){
    struct gfifo_s *gfifo;
    int tab[X] = {12, 13, 14, 15, 16};
    int i;
    void *deleted;

    gfifo = gfifo_new();

    for (i = 0; i < X; i++){
        gfifo_enqueue(gfifo, &tab[i]);
        gfifo_apply(gfifo, printg_int);
        putchar('\n');
    }

    printf("size : %d\n\n", gfifo_size(gfifo));

    for (i = 0; i < X; i++){
        gfifo_dequeue(gfifo, &deleted);
        printf("delete : %d \n", *(int *)deleted);
        gfifo_apply(gfifo, printg_int);
        putchar('\n');
    }

    gfifo_apply(gfifo, printg_int);
    putchar('\n');
    gfifo_del(gfifo);
}

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

Utiliser les fonctions des bibliothèques pour masquer l'implémentation.


```c
#include "gfifo.h"
#include "ififo.h"
#include <stdio.h>


ififo_s *ififo_merge(ififo_s *f1, ififo_s *f2){
    int nombre;
    ififo_s *ififo = ififo_new();

    while (f1 -> suivant != NULL || f2 -> suivant != NULL){
        if (f1 -> suivant == NULL){
            ififo_dequeue(f2, &nombre);
            ififo_enqueue(ififo, nombre);
        }
        else if (f2-> suivant == NULL){
            ififo_dequeue(f1, &nombre);
            ififo_enqueue(ififo, nombre);
        }
        else{
            if (f1-> suivant -> nombre > f2 -> suivant -> nombre){
                ififo_dequeue(f2, &nombre);
                ififo_enqueue(ififo, nombre);
            }
            else{
                ififo_dequeue(f1, &nombre);
                ififo_enqueue(ififo, nombre);
            }
        }
    }
    return ififo;
}

void trier(struct gfifo_s *gfifo){
    ififo_s *fifo1, *fifo2;
    ififo_s *sort;

    if (gfifo-> suivant != NULL){
        while (gfifo-> suivant -> noeud != NULL){
            gfifo_dequeue(gfifo, &fifo1);
            gfifo_dequeue(gfifo, &fifo2);
            sort = ififo_merge(fifo1, fifo2);

            gfifo_enqueue(gfifo, sort);
        }
    }
}

void test_merge(){
    printf("test merge \n");
    ififo_s *fifo_test, *fifo1, *fifo2;
    fifo1 = ififo_new();
    fifo2 = ififo_new();

    int i;
    int tab[5] = {1, 2, 3, 4, 5};
    int tab2[5] = {1, 3, 5, 15, 20};
    for (i = 0; i < 5; i++){
        ififo_enqueue(fifo1, tab[i]);
        ififo_enqueue(fifo2, tab2[i]);
    }

    fifo_test = ififo_merge(fifo1, fifo2);
    ififo_apply(fifo_test, print_int);

    ififo_del(fifo_test);
    ififo_del(fifo1);
    ififo_del(fifo2);
}
```

ififo_is_empty(..)

ififo_head(..)

gfifo_size(..) ==...

Cette fonction doit prendre en entrée une ififo et répartir ses éléments dans une gfifo + libérer les files intermédiaires vidées par ififo_merge()


```c
int main(void){
    test_merge();
    putchar('\n');

    int x = 0;
    int nb;
    gfifo_s *gfifo = gfifo_new();
    ififo_s *fifo;

    printf("Entrez 10 entiers :) \n");

    while (x < 10){
        printf("Entrez un entier : ");
        scanf("%d", &nb);
        fifo = ififo_new();
        ififo_enqueue(fifo, nb);
        gfifo_enqueue(gfifo, fifo);
        x++;
    }
    trier(gfifo);
    ififo_apply(gfifo -> suivant -> nombre, print_int);
    gfifo_del(gfifo);
    ififo_del(fifo);
    return 0;
}
```

écriture les entrées au clavier dans les tests (non reproductibles)

1 new/itération → fuite de mémoire.

1 seul del

```c
#include "files_entier.h"
#include "files_generique.h"
#define X 5

ififo_s *ififo_merge(ififo_s *f1, ififo_s *f2){

    ififo_s *ififo;
    int nb;

    ififo = ififo_new();

    while (f1 -> dernier != NULL){
        ififo_dequeue(f1, &nb);
        ififo_enqueue(ififo, nb);
    }

    while (f2 -> dernier != NULL){
        ififo_dequeue(f2, &nb);
        ififo_enqueue(ififo, nb);
    }

    ififo_del(f1);
    ififo_del(f2);

    return ififo;
}

gfifo_s *trier(struct ififo_s *f){
    gfifo_s *gfifo = (struct gfifo_s *)f;

    gfifo_apply(gfifo, printg_int);
    putchar('\n');
    return NULL;
}

int main(void){
    int i;
    struct ififo_s *fifo1, *fifo2, *fifoMerge;
    struct gfifo_s *gfifo;

    fifo1 = ififo_new();
    fifo2 = ififo_new();
    gfifo = gfifo_new();

    for (i = 1; i < X + 1; i++){
        ififo_enqueue(fifo1, i);
        ififo_enqueue(fifo2, i + 5);
    }

    puts("FIFO 1: ");
    ififo_apply(fifo1, print_int);
    putchar('\n');

    puts("FIFO 2: ");
    ififo_apply(fifo2, print_int);
    putchar('\n');

    puts("FIFO MERGE: ");
    fifoMerge = ififo_merge(fifo1, fifo2);
    ififo_apply(fifoMerge, print_int);
    putchar('\n');

    puts("FIFO TRIER: ");
    gfifo = trier(fifoMerge);
    gfifo_apply(gfifo, printg_int);
    putchar('\n');

    return 0;
}
```

```c
#include <stdlib.h>
#include <stdio.h>
#ifndef FILES_ENTIER
#define FILES_ENTIER

/* QUESTION 1 : */

typedef struct ififo_node_s ififo_node_s;
struct ififo_node_s
{
    int nombre;
    ififo_node_s *noeud;
};

/* QUESTION 2 : */

typedef struct ififo_s ififo_s;
struct ififo_s
{
    ififo_node_s *suivant;
    ififo_node_s *dernier;
};

ififo_s *ififo_new();
int ififo_is_empty(ififo_s *f);
int ififo_enqueue(ififo_s *f, int nb);
int ififo_dequeue(ififo_s *f, int *nb);
int ififo_head(const ififo_s *f);
typedef void(func_t)(int);
int ififo_apply(ififo_s *f, func_t *fn);
void ififo_del(ififo_s *f);
void print_int(int i);

#endif
```

```c
#include <stdio.h>
#include <stdlib.h>
#ifndef FILES_GENERIQUE
#define FILES_GENERIQUE

/* QUESTION 1 : */

typedef struct gfifo_node_s gfifo_node_s;

struct gfifo_node_s{
    void *nombre;
    struct gfifo_node_s *noeud;
};

typedef struct gfifo_s gfifo_s;

struct gfifo_s{
    struct gfifo_node_s *suivant;
    struct gfifo_node_s *dernier;
};

gfifo_s *gfifo_new();
int gfifo_del(gfifo_s *f);

int gfifo_size(gfifo_s *f);

int gfifo_enqueue(gfifo_s *f, void *nb);
int gfifo_dequeue(gfifo_s *f, void **nb);

typedef void(gfunc_t)(void *);
void printg_int(void *i);
int gfifo_apply(gfifo_s *f, gfunc_t *fn);

#endif
```

pas nécessairement un nombre

suivant

premier de la file

pas toujours un nombre -

Attention au nommage des champs/variables !...

```c
#include <stdlib.h>
#include <stdio.h>
#ifndef GFIFO
#define GFIFO

typedef struct gfifo_node_s gfifo_node_s;

struct gfifo_node_s{
    void *nombre;
    gfifo_node_s *noeud;
};

typedef struct gfifo_s gfifo_s;

struct gfifo_s{
    gfifo_node_s *suivant;
    gfifo_node_s *dernier;
};

struct gfifo_s *gfifo_new();

int gfifo_del(struct gfifo_s *f);

int gfifo_size(struct gfifo_s *f);

int gfifo_enqueue(struct gfifo_s *f, void *p);
int gfifo_dequeue(struct gfifo_s *f, void **p);
void print_int(int i);

typedef void(gfunc_t)(void *);
int gfifo_apply(struct gfifo_s *f, gfunc_t *fn);

#endif
```

```c
#include <stdlib.h>
#include <stdio.h>
#ifndef IFIFO
#define IFIFO

typedef struct ififo_node_s ififo_node_s;

struct ififo_node_s{
    int nombre;
    ififo_node_s *noeud;
};

typedef struct ififo_s ififo_s;

struct ififo_s{
    ififo_node_s *suivant;
    ififo_node_s *dernier;
};

struct ififo_s *ififo_new();

int ififo_is_empty(struct ififo_s *f);

int ififo_enqueue(struct ififo_s *f, int new_val);

int ififo_dequeue(struct ififo_s *f, int *head);

int ififo_head(const struct ififo_s *f);

typedef void(func_t)(int);

int ififo_apply(struct ififo_s *f, func_t *fn);

void ififo_del(struct ififo_s *f);

#endif
```