```c
#include <stdio.h>
#include <stdlib.h>
/*
 * FIFO Listes chainÃ©es
 * Slide 18 ou 19.
 *
 * Start with structs.
 * Them simple main.
 * Then simple functions !!!
 * Then complex ones.
 * Then go back to main.
 */
typedef unsigned char val_type;

struct cellule {
        val_type val;
        struct cellule *suiv;
};
struct fifo {
        struct cellule *tete;
        struct cellule *queue;
};

int fifo_vide(struct fifo file) {
        return (file.tete == NULL);
}

int fifo_pleine(struct fifo file) {
        return 0;
}

void print_fifo(struct fifo file) {
        struct cellule *tmp = file.tete;

        printf("-----------\n");
        printf("fifo_pleine ? %d, fifo_vide ? %d :::: ", fifo_pleine(file), fifo_vide(file));
        /*printf("Fifo : ");*/
        while (tmp != NULL) {
                printf("%c,", tmp->val);
                tmp = tmp->suiv;
        }
        /*printf("\nFifo est vide ? %d, Fifo est pleine ? %d\n",
         * fifo_vide(file), fifo_pleine(file));*/
        printf("\n----------\n\n");
}


void init_fifo(struct fifo *ptr_file)
{
        ptr_file->tete = NULL;
        ptr_file->queue = NULL;
}

val_type first(struct fifo file)
{
        return file.tete->val;
}

val_type get(struct fifo *ptr_file)
{
        val_type v = ptr_file->tete->val;

        struct cellule *tmp = ptr_file->tete;

        ptr_file->tete = ptr_file->tete->suiv;
        if (ptr_file->tete == NULL)
                ptr_file->queue = NULL;

        free(tmp);

        return v;
}

// put = ajouter en queue !
void put(struct fifo *ptr_file, val_type val) {
        struct cellule *tmp = malloc(sizeof(struct
cellule));
        tmp->val = val;
        tmp->suiv = NULL;

        if (ptr_file->tete == NULL)
                ptr_file->tete = tmp;
        else
                ptr_file->queue->suiv = tmp;

        ptr_file->queue = tmp;
}

```

```
int simple_main() {
        struct fifo f;
        init_fifo(&f);

        print_fifo(f);

        val_type x = 'W';
        put(&f, x);
        print_fifo(f);

        printf("Fifo vide 1 = %d ?\n", fifo_vide(f));

        x = get(&f);
        print_fifo(f);

        printf("Fifo vide 2 = %d ?\n", fifo_vide(f));

        return 0;
}

int main_complex() {
        struct fifo f;
        init_fifo(&f);

        for (val_type c = 'a'; c < 'a' + 26; c++) {
                print_fifo(f);
                put(&f, c);
        }

        while (!fifo_vide(f)) {
                printf("RÃ©cupÃ©rÃ©: %c\n", get(&f));
                // get(&f);
                print_fifo(f);
        }

        return 0;
}
int main(void) {
        simple_main();
        return 0;
}

/* //UNUSED because get function is there !!!
void sup_tete(struct fifo * ptr_file) {
        struct cellule * tmp = ptr_file->tete ;

        if( ptr_file->tete == ptr_file->queue) {
                ptr_file->tete  = NULL;
                ptr_file->queue = NULL;
        } else {
                ptr_file->tete = ptr_file->tete->suiv ;
        }
        free(tmp);
}
*/
//UNUSED !!!
void sup_tete_fifo(struct fifo * ptr_file) {
        struct cellule * tmp = ptr_file->tete ;

        if( ptr_file->tete == ptr_file->queue) {
                ptr_file->tete  = NULL;
                ptr_file->queue = NULL;
        } else {
                ptr_file->tete = ptr_file->tete->suiv ;
        }
        free(tmp);
}

//DANS DS... pas donner aux eleves
int delete_queue(struct fifo * f ) {
        if(f==NULL)       return 0; //Nothing deleted, no fifo
        if(f->tete==NULL) return 0; //Nothing deleted, fifo empty

        if(f->tete->suiv == NULL) { //Only one cell, update tete et queue
                sup_tete_fifo(f);
                return 1;               //deleted a value.
        }
        struct cellule * tmp = f->tete;
        while(tmp->suiv->suiv != NULL)
                tmp=tmp->suiv;
        //tmp pointe sur l'avant dernier
        free(tmp->suiv);  //free last
        tmp->suiv = NULL; //update pointer
        f->queue = tmp;   //update tail
        return 1;
}
```