```c
typedef struct contact_t contact_t;
typedef struct agenda_t agenda_t;
struct contact_t {
  char * nom, * prenom, * phone;
  struct contact_t * suivant;
};
struct agenda_t {
  struct contact_t * premier, * dernier;
};
extern contact_t * addContact       (agenda_t * a);
extern void        setContactNom    (contact_t * c, char * nom);
extern void        setContactPrenom (contact_t * c, char * prenom);
extern void        setContactPhone  (contact_t * c, char * phone);
extern void        printAgenda      (agenda_t * a);
extern void        freeAgenda       (agenda_t * a);
```

```c
#include "agenda.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
extern contact_t * addContact(agenda_t * a) {
  contact_t * c;
  if(a->premier == NULL)
    c = a->premier = a->dernier = malloc(sizeof c[0]);
  else {
    c = a->dernier->suivant = malloc(sizeof c[0]);
    a->dernier = a->dernier->suivant;
  }
  if(c == NULL) {
    fprintf(stderr, "Erreur d'allocation memoire\n");
    exit(1);
  }
  c->nom = c->prenom = c->phone = NULL;
  c->suivant = NULL;
  return c;
}
extern void setContactNom(contact_t * c, char * nom) {
  c->nom = malloc((strlen(nom) + 1) * sizeof c->nom[0]);
  if(c->nom == NULL) {
    fprintf(stderr, "Erreur d'allocation memoire\n");
    exit(1);
  }
  strcpy(c->nom, nom);
}
extern void setContactPrenom(contact_t * c, char * prenom) {
  c->prenom = malloc((strlen(prenom) + 1) * sizeof c->prenom[0]);
  if(c->prenom == NULL) {
    fprintf(stderr, "Erreur d'allocation memoire\n");
    exit(1);
  }
  strcpy(c->prenom, prenom);
}
extern void setContactPhone(contact_t * c, char * phone) {
  c->phone = malloc((strlen(phone) + 1) * sizeof c->phone[0]);
  if(c->phone == NULL) {
    fprintf(stderr, "Erreur d'allocation memoire\n");
    exit(1);
  }
  strcpy(c->phone, phone);
}
extern void printAgenda(agenda_t * a) {
  contact_t * c = a->premier;
  int nbContacts = 0;
  while(c != NULL) {
    nbContacts++;
    printf("Nom : %s\nPrenom : %s\nPhone : %s\n", c->nom, c->prenom, c->phone);
    printf("*********************************************************\n");
    c = c->suivant;
  }
  printf("\n*** Vous avez %d entree(s) dans votre agenda ***\n", nbContacts);
}
extern void freeAgenda(agenda_t * a) {
  contact_t * c = a->premier, * tmp;
  while(c != NULL) {
    free(c->nom); free(c->prenom); free(c->phone);
    tmp = c; c = c->suivant;
    free(tmp);
  }
}
```

FIG. 1 – agenda.h et agenda.c

```c
#include "agenda.h"
#include <stdio.h>
#include <stdlib.h>
static void addEntree(agenda_t * a) {
  char tmp[256];
  contact_t * c;
  c = addContact(a);
  printf("Entrez le nom : ");    fgets(tmp, sizeof tmp, stdin); setContactNom(c, tmp);
  printf("Entrez le prenom : "); fgets(tmp, sizeof tmp, stdin); setContactPrenom(c, tmp);
  printf("Entrez le tel : ");    fgets(tmp, sizeof tmp, stdin); setContactPhone(c, tmp);
}
int main(void) {
  agenda_t monAgenda = {NULL, NULL};
  char c , reste;
  while(1) {
    printf("cmd : ");
    reste = c = getchar(); while(reste != '\n') reste = getchar();
    switch(c) {
    case 'c': /* Consulter l'agenda */
      printAgenda(&monAgenda);
      break;
    case 'a': /* Ajouter une entree dans l'agenda */
      addEntree(&monAgenda);
      break;
    case 'h': /* Afficher l'aide */
      printf("c : consulter l'agenda\n");
      printf("a : ajouter une entree dans l'agenda\n");
      printf("h : aide\n");
      printf("q : quitter le programme\n");
      break;
    case 'q': /* Quitter le programme */
      freeAgenda(&monAgenda);
      exit(0);
    default: /* Traiter les autres cas */
      printf("La commande 'h' permet d'obtenir l'aide\n");
      break;
    }
  }
  return 0;
}
```

FIG. 2 – main.c

Exercice :

- En ajoutant un nouveau contact, nous entrons une chaine vide pour **nom**, **prenom** ou **tel** (CTRL-D). Un bug peut apparaître à l'affichage du contact que nous venons de rajouter. Expliquer pourquoi et corriger ce bug (voir la fonction `addEntree`) ;
- Créer une fonction `setContactField(char ** dst, char * src);` qui remplace : `setContactNom`, `setContactPrenom` et `setContactPhone`. Supprimer ces dernières et modifier les appels de fonction ;