INFO0946 **Examen INFO0946** Juin 2020	NOM : . . . . . PRÉNOM : . . . . . Matricule ULiège : ☐☐☐☐☐☐☐☐ **Remplissez ce cadre en MAJUSCULES et LISIBLEMENT, svp**

Introduction à la Programmation (durée : 3h00)

Question	**Points**
Question 1	. . . . . /10
Question 2	. . . . . /20
Question 3	. . . . . /20
Consignes	

Consignes (à lire attentivement avant de répondre aux questions)

Suite à la crise sanitaire liée à l'épidémie de Covid-19, l'examen est organisé à distance via une session WebEx (pour poser des questions sur le questionnaire), via CAFÉ pour l'encodage de vos réponses (ainsi que la correction automatique) et la Plateforme de Soumission pour déposer vos réponses.

1. Le présent questionnaire et les différents fichiers annexes (e.g., les fichiers servant de canevas pour la soumission de vos réponses) sont disponibles sur eCampus (INFO0946, Sec. Examen).
2. Contrairement aux Challenges, CAFÉ ne fera pas de correction automatique avec feedback sur vos exercices. La seule chose que vérifiera CAFÉ, c'est l'encodage de vos réponses dans le canevas. Un email vous sera envoyé automatiquement avec le résultat de ces tests. Si jamais il y a des erreurs d'encodage, vous avez la possiblité de charger de nouveaux votre fichier (autant de fois que nécessaire, dans les limites du temps imparti par l'examen).
3. Il y a un fichier de canevas par question. Vous devez donc télécharger, depuis eCampus (Sec. Examen), trois fichiers textes (`question1.txt`, `question2.txt`, `question3.txt`). Ces fichiers doivent être complétés et placés dans une archive `zip` pour la soumission (voir page suivante).
4. La correction de l'examen sera réalisée automatiquement par CAFÉ, après l'examen en lui-même. Il est, dès lors, important que chacun de vos codes compile. Sinon, vous risquez une note nulle à la question ! !
5. La durée de l'examen est de 3h. N'attendez pas la dernière seconde pour soumettre votre fichier.

INFO0946 NOM : ..... PRÉNOM : .....

Soumission

Pour chacune des questions de cet examen, vous disposez, sur eCampus (Sec. Examen), d'un fichier texte (`question1.txt`, `question2.txt`, `question3.txt`) à compléter.

Vous devez soumettre ces trois fichiers textes, sous la forme d'une archive `zip`, sur la Plateforme de Soumission. Voici comment procéder sur les systèmes d'exploitation les plus courants. **Placez les trois fichiers à la racine de l'archive (donc pas dans un dossier).**

Sous Windows Il suffit de cliquer sur le fichier à l'aide du bouton droit de la souris, sélectionner « Envoyer vers... » et sélectionner ensuite « Dossier compressé ».

Sous Linux (Ubuntu, Fedora, Linux Mint, ...) Il suffit de cliquer sur le fichier à l'aide du bouton droit de la souris, sélectionner « Compresser... ». Veillez bien à sélectionner « `.zip` » dans la liste des extensions possibles pour le fichier.

Sous OS X Cliquez sur le fichier en maintenant la touche Contrôle enfoncée (ou cliquez avec 2 doigts), sélectionnez « Compresser ».

Dans tous les cas Ne soumettez pas de fichier `.tar.gz`, `.7z`, `.rar` ou autre ! C'est bien un fichier `.zip` qui est attendu. Le nom de l'archive importe peu, tant que c'est une archive `zip` valide, dont le nom se termine bien par « `.zip` » **et ne comporte pas de caractères spéciaux comme des espaces, des parenthèses, etc.**

Question 1 : Pointeurs ($10pts$)

	240	27
`data:`	236	108
	⋮	⋮
	120	3
	116	17
	112	5
	108	120
`x:`	104	236

La première colonne donne le nom des variables, la deuxième donne des adresses mémoires (nous les exprimerons dans cette question en **base 10**) et, enfin, la troisième donne la valeur stockée à cette adresse (`??` signifie que la valeur est indéterminée). De plus, voici comment ont été déclarées les variables (on suppose que les entiers – `int` – sont représentés sur 4 octets [1]) :

```
int data, x, *ptr = &x;
char *E = "Examen", *F = E + 6;
```

Dans cette question, vous devez indiquer la valeur des **expressions** suivantes. Considérez qu'entre chaque expression, la mémoire est réinitialisée telle que présentée dans le schéma. Si un accès mémoire est demandé à une adresse hors de l'intervalle $[104, 240]$, mentionnez l'erreur de segmentation par la valeur `SF` [2].

1. `data`
2. `*data`
3. `**data`
4. `&data`
5. `&*data`
6. `*&x`
7. `*ptr++`
8. `++&data`
9. `*++ptr`
10. `ptr + data`
11. `&data - ptr`
12. `(short *) ptr + x`
13. `x & *ptr`
14. `*F`
15. `F - E`
16. `**F`
17. `*(x = data)`
18. `*(x == data)`
19. `F[-2] - *(E + 2)`
20. `E[E[5] - E[**data]] - 'x'`

Lors de votre soumission, vous devrez indiquer les valeurs de ces expressions, en fonction de l'état de la mémoire décrit ci-dessus. La façon de formuler vos réponses est indiquée dans le fichier servant de canevas à votre soumission, fichier disponible sur eCampus (Sec. Examen – `question1.txt`).

1. les `short` sur 2, les `long sur 8`
2. Pour `S`egmentation `F`ault.

Question 2 : Construction par Invariant ($20pts$)

On dispose de deux tableaux d'entiers `A` et `B` (de dimensions respectives `N` et `M`[3]). Les tableaux `A` et `B` sont triés par ordre strictement croissant (i.e., du plus petit au plus grand).[4] Dans cette question, nous vous demandons de fusionner les éléments communs aux tableaux `A` et `B` dans un troisième tableau, `C`, trié lui aussi par ordre strictement croissant.[5] La taille du tableau `C` est toujours suffisamment grande.

Ainsi, par exemple, Si `N` vaut 5 et `M` vaut 7, les tableaux `A` et `B` peuvent ressembler à ceci :

	0				4
A :	3	5	6	9	10

	0						6
B :	2	3	4	6	7	8	9

Le tableau `C` résultant de la fusion serait le suivant (le `?` désignant une valeur indéterminée dans une case du tableau `C`) :

	0											11
C :	3	6	9	?	?	?	?	?	?	?	?	?

Attention, pour résoudre ce problème, vous devez respecter ces contraintes :

— vous ne pouvez utiliser qu'une et une seule boucle de type `while` ;

— vous devez viser une complexité théorique $\mathcal{O}(M+N)$. De plus, vous ne pouvez entrer dans votre boucle, qu'au plus $M+N$ fois. Attention, le nombre d'itérations sera compté !

— Rien ni personne ne vous demande d'écrire quoi que ce soit à l'écran : **ne polluez donc pas la sortie standard sous peine d'empêcher la correction de votre travail !**

Le squelette de votre code est le suivant :

```c
#include <stdio.h>

int main(){
  const unsigned int N = ...;
  const unsigned int M = ...;
  const unsigned int L = ...; /* garanti suffisamment grand */
  int A[N];
  int B[M];
  int C[L];

  /*
   * Code de remplissage des tableaux A et B.
   * Ce code ne vous est pas donne.
   */

  // Votre code viendra ici (variables + instructions)

}//fin programme
```

Vous devez considérer, lors de la soumission, que votre extrait de code est copié et collé[6] à l'endroit indiqué par le commentaire « `Votre code viendra ici (variables + instructions)` ». Dès lors, redéclarer les constantes `N`, `M`, ou `L` ou encore les tableaux `A`, `B` ou `C` fera échouer la compilation.

Lors de votre soumission, vous devrez fournir

1. l'Invariant de votre boucle ;
2. la Fonction de Terminaison de votre boucle ;

3. `N` et `M` peuvent être nuls.
4. Les éléments dans un tableau sont forcément uniques.
5. Les éléments dans le tableau `C` sont forcément uniques.
6. En fait, c'est à quelques détails près le cas.

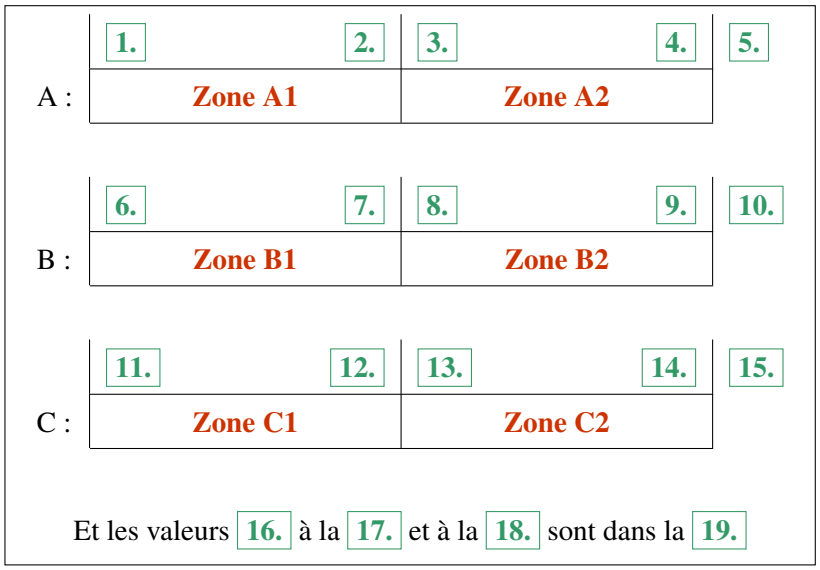
3. le code C.

La façon de formuler votre Invariant et votre Fonction de Terminaison est indiqué dans les sections ci-dessous. Le fichier servant de canevas à votre soumission est disponible sur eCampus (Sec. Examen, – `question2.txt`). En outre, le fichier `question2.c` est disponible sur eCampus (Sec. Examen). Il vous permet de tester la compilation de votre code (sans devoir tout réécrire par vous-mêmes). Attention, les valeurs données pour `N`, `M`, `L`, `A` et `B` données dans `question2.c` sont des cas particuliers.

Question 2.A : Invariant

Ce problème peut être résolu sans découpe en sous-problème.

Voici un Invariant muet [7] :



Selon l'équation fondamentale des Invariants des codes qui manipulent des tableaux, i.e. :

$$1 \text{ Tableau} = 1 \text{ Dessin},$$

cet Invariant doit représenter les trois tableaux manipulés par votre code. Les boites 1. à 15. servent à mentionner des indices de ces tableaux. Ces boites doivent donc être remplacées par des constantes ou des noms de variables.

Toutes les boites ne doivent pas forcément être précisées : si vous indiquez que la valeur de la boite 9. est « k », il est implicite que la valeur de la boite 10. est est « $k + 1$ » : inutile alors de le préciser.

Pour les autres boites, les choix disponibles sont résumés dans la table 1.

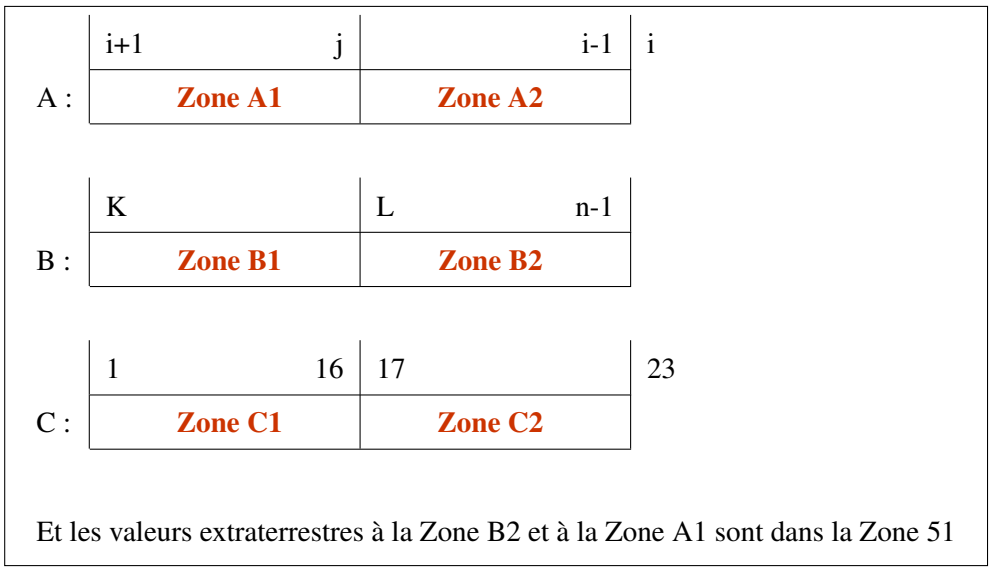
Veuillez indiquer, dans le fichier de soumission, à la réponse sur l'Invariant, le numéro de la boite, suivi d'un point, suivi de la valeur numérique ou du nom de variable ou constante de votre choix (pour les boites de 1 à 15) ou d'un nombre entre 1 et 7 correspondant à votre choix (pour les boites 16 à 19).

7. Tout ce qui est dans le cadre fait partie de l'Invariant

Pour la boite 16	**Pour les boites 17, 18 et 19**
1. différentes ;	1. Zone A1
2. maximales ;	2. Zone A2
3. parcourues ;	3. Zone B1
4. analysées ;	4. Zone B2
5. singulières ;	5. Zone C1
6. extraterrestres ;	6. Zone C2
7. communes.	7. Zone 51

TABLE 1 – Invariant : Possibilités de choix pour les boites **16** à **19**.

Si vous pensez qu'un bon invariant serait :



Et les valeurs extraterrestres à la Zone B2 et à la Zone A1 sont dans la Zone 51

Alors, complétez le fichier de réponse comme suit :

```
1. i+1
2. j
3. _
4. i-1
5. i
6. K
7.
8. L
9. n-1
10.
11. 1
12. 16
13. 17
14.
15. 23
16. 6
17. 4
18. 1
19. 7
```

Vous voyez qu'il est possible d'affecter une variable ou une constante d'un modificateur +1 ou -1. Il suffit dans ce cas d'écrire ce +1 ou -1 comme montré ci-dessus. N'introduisez pas de parenthèses.

Dans tous les cas, si vous pensez que la bonne réponse consiste à ne rien écrire à l'emplacement d'une boite, n'inscrivez rien comme réponse ou un « _ ».

Pour votre facilité, (et pour les distraits qui en oublieraient un), les numéros des 19 boites sont déjà inscrits dans le fichier de réponse.

Question 2.B : Fonction de Terminaison

Dand cette question, nous vous demandons de fournir la Fonction de Terminaison de votre boucle.

Dans le fichier de réponses, à la question sur la Fonction de Terminaison, nous vous demandons de nous la fournir sous la forme d'une **expression C** valide.

La correction s'assurera entre autres :
— Que cette expression utilise des variables de votre code ;
— Que son domaine est bien l'ensemble des Entiers (positifs si le gardien est vrai) ;
— Que sa valeur décroit strictement entre deux itérations.

Si vous pensez que la Fonction de Terminaison de votre code, qui manipule la variable `toto` et la constante `G` devrait être :

$$t = \texttt{G} + 17 - \texttt{toto}$$

Encodez :

```
G + 17 - toto
```

N'indiquez pas « `F =` » ou « `t =` » mais seulement l'expression qui sert à calculer la valeur de votre Fonction de Terminaison. Il est inutile de rappeler que votre expression entière est positive en rajoutant un « > 0 » (en plus, cela transforme votre expression en prédicat et cela rend votre réponse incorrecte).

Question 3 : Programme Complet ($20pts$)

Ces dernières années, l'Institut Montefiore s'est vue développer des élèvages de chats afin d'abreuver Facebook de nombreuses vidéos sans le moindre intérêt. Afin de débuter au mieux l'élevage, vous commencez à rédiger une application permettant de gérer la *chatterie* [8].

Un squelette de programme, basé sur les principes de la programmation séparée, a été créé pour vous. Il est composé de deux fichiers : le header (`question3.h`) et le module (`question3.c`). Ces deux fichiers sont disponibles sur eCampus (Sec. Examen). Votre objectif est de compléter le code existant et de soumettre vos réponses dans le fichier texte correspondant (`question3.txt`, disponible sur eCampus, Sec. Examen).

1. `question3.h`. C'est le fichier d'en-tête contenant les déclarations des structures de données et les prototypes des différentes fonctions et procédures. L'extrait de code 1 vous donne son contenu.
2. `question3.c`. Il s'agit de l'implémentation du header `question3.h`. Les fonctions `charger_chatons()`, `cree_chat()`, `date_du_jour()` et `date_vers_chaine()` ont déjà été implémentées. Vous devez donc les utiliser quand vous le jugerez nécessaire. L'extrait de code 2 vous donne le squelette du fichier `chat.c`. En outre, le fichier `question3.c` contient une fonction `main()` reprenant le programme principal. Cette fonction `main()` est importante pour la Question 3.C.

Extrait de code 1 – Fichier `question3.h`

```c
#ifndef __QUESTION3__
#define __QUESTION3__

#include <stdio.h>
#include <stdlib.h>
#include <assert.h>

//défintion du type pour une date
typedef struct{
  unsigned short jour;
  unsigned short mois;
  unsigned short annee;
}Date;

//définition d'un chat
typedef struct{
  char *nom;
  char sexe; //'F' ou 'M'
  int vaccine; //1 si le chat est vacciné, 0 sinon
}Chat;

//définition d'une portée de chat
typedef struct{
  Chat *pere;
  Chat *mere;
  Chat *chatons;
  unsigned int nb_chatons;
  Date date_naissance;
}Portee;

//définition d'une chatterie
typedef struct{
  Portee *portees;
  int nb_portees;
}Chatterie;

/*
 * Charge, depuis un fichier dont le nom est contenu dans nom_fichier, tous les
 * chatons de la portée et les place dans la portée pointée par p.
```

8. Une chatterie est un lieu où sont élevés des chats de race.

```
 *
 * @pré: p pointe vers une portée valide, p->chatons doit exister, nb_chatons>0,
 *       nom_fichier valide
 *
 * @post: retourne 1 si tous les chatons ont pu être chargé dans la portée.
 *        -1 en cas de problème.
 */
int charger_chatons(char *nom_fichier, Portee *p, int nb_chatons);

/*
 * Crée un chat dont le nom est contenu dans nom.
 *
 * @pré: nom_fichier est valide, sexe \in {'F', 'M'}
 *
 * @post: retourne un pointeur vers un chat valide.
 *        NULL en cas de problème.
 */
Chat *cree_chat(char *nom, char sexe, int vaccine);

/*
 * Transforme une date en chaîne de caractères au format jj/mm/aaaa.
 *
 * @pre: d est une date valide.
 * @post: chaîne de caractères représentant la date.  Ne rate jamais.
 */
char *date_vers_chaine(Date *d);

/*
 * Question 3.A
 *
 * Crée une portée.  attention, les parents sont déjà vaccinés.
 */
Portee *cree_portee(char *nom_pere, char *nom_mere, int nb_chatons,
                    char *nom_fichier, Date date_naissance);

/*
 * Queston 3.B
 *
 * Sauve, dans un fichier donné, tous les chatons d'une portée donnée.
 * Le format de la sauvegarde est le suivant:
 * date_de_naissance, père, mère
 * nom_chaton1, sexe, {vacciné, pas vacciné}
 * nom_chaton2, sexe, {vacciné, pas vacciné}
 * ...
 * nom_chatonn, sexe, {vacciné, pas vacciné}
 */
int sauve_portee(Portee *p, char *nom_fichier);

/*
 * Question 3.C
 *
 * Détermine, pour toutes les portées, le nombre de chatons déjà vaccinés, le
 * nombre de chatons mâle et le nombre de chatons femelle.
 */

#endif
```

Extrait de code 2 – Fichier `question3.c`

```
#include "question3.h"

/*
 * Variables globales permettant l'implémentation rapide de certaines
 * fonctionnalités non fournies
```

```
 */
Chat test = {"Poupousse", 'M', 1};
Chat chatons[10] = {{"Chat1", 'M', 1},{"Chat2", 'M', 1},{"Chat3", 'M', 1},
                    {"Chat4", 'M', 1},{"Chat5", 'M', 1},{"Chat6", 'M', 1},
                    {"Chat7", 'M', 1},{"Chat8", 'M', 1},{"Chat9", 'M', 1},
                    {"Chat10", 'M', 1}};
Portee test_p = {&test, &test, chatons, 10, {9,6,2020}};
Chatterie test_c = {&test_p, 1};

int charger_chatons(char *nom_fichier, Portee *p, int nb_chatons){
  p->chatons[0] = chatons[0];
  p->chatons[1] = chatons[1]; // Incorrect mais permet les tests.
  return 1;
}//fin charger_chatons()

Chat *cree_chat(char *nom, char sexe, int vaccine){
  return &test; // Incorrect mais permet les tests.
}//fin cree_chat()

char *date_vers_chaine(Date *d){
  return "09/06/2020"; // Incorrect mais permet les tests.
}//fin date_vers_chaine()

Portee *cree_portee(char *nom_pere, char *nom_mere, int nb_chatons,
                    char *nom_fichier, Date date_naissance){
  //VOTRE CODE ICI
}//fin cree_portee()

int sauve_portee(Portee *p, char *nom_fichier){
  //VOTRE CODE ICI
}//fin sauve_portee()

//QUESTION 3.c

static Chatterie *cree_chatterie(int nb_portees){
  return &test_c; // Incorrect mais permet les tests.
}//fin cree_chatterie()

int main(){
  Chatterie *montef;
  unsigned int nb_vaccine, nb_male, nb_femelle;

  montef = cree_chatterie(1);
  if(montef==NULL)
    return -1;

  /*
   * STATISTIQUES sur la chatterie (utilisation de la fonction/procédure de
   * la Question 3.C)
   * On veut afficher à l'écran le nombre de  chatons encore à vacciner, le
   * nombre de femelles et de mâles.
   * Les variables nécessaires ont déjà été déclarées dans le main().  Vous ne
   * pouvez utiliser rien d'autre.
   */
  //L'invocation de votre fonction vient ici
  printf("Nombre de chatons à encore vacciner: %u\n", nb_vaccine);
  printf("Nombre de femelles: %u\n", nb_femelle);
  printf("Nombre de mâles: %u\n", nb_male);

  return 0;
}//fin programme
```

Dans toutes les sous-questions et quand cela s'avère nécessaire, il est demandé d'appliquer les principes de la programmation défensive.

En outre, vous pouvez toujours invoquer n'importe quelle fonctionnalité de la chatterie (cf. `question3.h`), même si vous ne l'avez pas implémentée.

Question 3.A

Implémentez la fonction `Portee *cree_portee(char *nom_pere, char *nom_mere, int nb_chatons, char *nom_fichier, Date date_naissance);`. Cette fonction crée une portée comportant `nb_chatons` chatons et dont le père s'appelle `nom_pere` et la mère `nom_mere`.[9] La portée contient `nb_chatons`. La date de naissance des chatons est donnée. Les informations relatives à chaque chaton doivent être chargées depuis le fichier dont le nom est contenu dans `nom_fichier`. La fonction retourne `NULL` en cas de problème.

Insérez le corps de votre fonction dans le fichier `question3.txt`. Il est inutile d'ajouter des directives de prépro-cessing (`#include`, `#define`, etc.), de réécrire la signature de la fonction ainsi que les accolades représentant le bloc du corps de la fonction. Vous ne pouvez pas non plus redéclarer les paramètres formels de la fonction. Faites attention à ne pas utiliser l'underscore (_) dans vos noms de variables (surtout en début de nom : c'est interdit par le standard du langage C).

Question 3.B

Implémentez la fonction `int sauve_portee(Portee *p, char *nom_fichier);` dans le fichier `question3.c` Cette fonction sauve, dans un fichier donné, tous les chatons d'une portée donnée. Elle renvoie -1 en cas d'erreur, 0 si tout se passe bien. Le format du fichier est le suivant :

```
date_de_naissance, père, mère
nom_chaton, sexe, {vacciné, pas vacciné}
nom_chaton, sexe, {vacciné, pas vacciné}
...
nom_chaton, sexe, {vacciné, pas vacciné}
```

Par exemple, si une portée compte 2 chatons :

```
15 juin 1994, Simba, Nala
Kiara, F, vacciné
Kion, M, vacciné
```

Insérez le corps de votre fonction dans le fichier `question3.txt`. Il est inutile d'ajouter des directives de prépro-cessing (`#include`, `#define`, etc.), de réécrire la signature de la fonction ainsi que les accolades représentant le bloc du corps de la fonction. Vous ne pouvez pas non plus redéclarer les paramètres formels de la fonction. Faites attention à ne pas utiliser l'underscore (_) dans vos noms de variables (surtout en début de nom : c'est interdit par le standard du langage C) .

Question 3.C

Écrivez une fonction ou procédure dont le nom est `comptage` et qui retourne le nombre de chatons vaccinés, le nombre de femelles et le nombre de mâles pour une chatterie donnée.

Insérez le prototype et le corps de votre fonction ou procédure dans le fichier `question3.txt`. Il est inutile d'ajou-ter des directives de préprocessing (`#include`, `#define`, etc.). Faites attention à ne pas utiliser l'underscore (_) dans vos noms de variables (surtout en début de nom : c'est interdit par le standard du langage C).

9. Les parents de la portée sont vaccinés.

Question 3.D

Écrivez la ligne correspondant à l'appel de votre fonction ou procédure `comptage`, c'est-à-dire l'instruction qui doit remplacer le commentaire « `L'invocation de votre fonction vient ici` » à la ligne 60 de l'extrait de code 2.

Insérez l'instruction dans le fichier `question3.txt`. N'écrivez que l'instruction correspondant à l'invocation de votre fonction ou procédure `comptage`.