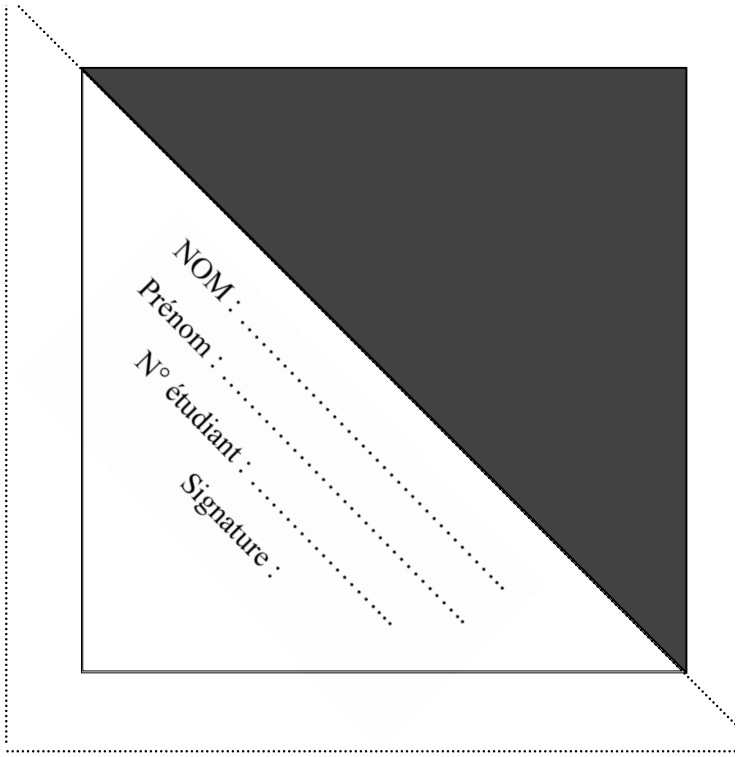
Université Claude Bernard Lyon 1
Licence Sciences, Technologies, Santé – L2
Année 2009-2010, 1er semestre

LIF5 – Algorithmique & Programmation procédurale

Contrôle à mi-parcours
16 novembre 2009



Durée : **1h30**

Documents et téléphones portables interdits.

Le barême est donné à titre indicatif.

Travaillez au brouillon d'abord de sorte à rendre une copie propre – il ne vous sera pas donné de copie supplémentaire si vous vous trompez.

Note :

Exercice 1 : Tri à bulles (6 points)

Le tri à bulles est un algorithme de tri qui s'appuie sur des permutations répétées d'éléments contigus qui ne sont pas dans le bon ordre.

Procédure tri_bulles(tab : tableau [1..n] de réels)
Précondition : tab est un tableau contenant n réels
Postcondition : les éléments de tab sont triés dans l'ordre croissant
Paramètre en mode donnée : aucun
Paramètre en mode donnée-résultat : tab
Variables locales : i, j : entiers, e : réel

```
Début
1  i ← 1
2  Tant que i <= n-1 Faire
3        j← n
4        Tant que j >= i+1 Faire
5              Si tab[j] < tab[j-1] Alors
6                    {on permute les deux éléments}
7                    e ← tab[j-1]
8                    tab[j-1] ← tab[j]
9                    tab[j] ← e
10                   Fin Si
11                   j ← j-1
12             Fin Tant que
13             i ← i + 1
14       Fin Tant que
Fin tri_bulles
```

1. Soit le tableau suivant : {53.8, 26.1, 2.5, 13.6, 8.8, 4.0}. Donnez les premiers états intermédiaires par lesquels passe ce tableau lorsqu'on lui applique la procédure tri_bulles : vous indiquerez l'état du tableau à chaque fois que j vient de changer de valeur, jusqu'à avoir complètement rempli la table ci-après (on ne vous demande pas d'aller jusqu'au tri complet du tableau).

Valeur de i	Valeur de j	Etat du tableau juste après que i ou j ait changé de valeur (donc au moment où l'on teste la condition du Tant que de la ligne 2 ou de la ligne 4)					
		tab[1]	tab[2]	tab[3]	tab[4]	tab[5]	tab[6]
1	6	53.8	26.1	2.5	13.6	8.8	4.0

2. Complétez la phrase suivante de sorte à ce qu'elle corresponde à l'invariant de boucle du Tant que interne (boucle sur j, lignes 4 à 12 de l'algorithme).

« Lorsqu'on vient de décrémenter j (ligne 11), le _____ du sous-tableau tab[_____... _____] se trouve en position _____. De plus, si j>1, les éléments du sous-tableau tab[_____..._____] occupent les mêmes positions qu'avant le démarrage de la boucle sur j. »

3. Déduisez-en la propriété que présente le tableau lorsqu'on a terminé cette boucle interne, c'est-à-dire lorsqu'on arrive sur la ligne 13.

4. Calculez le nombre total d'affectations de réels réalisées par la procédure tri_bulles lors du tri complet d'un tableau de *n* réels, dans le cas le plus défavorable.

Exercice 2 : Fusion de deux monotonies (6 points)

On dispose de deux fichiers contenant chacun une séquence triée de réels. En d'autres termes, chacun des deux fichiers contient une monotonie et une seule – il s'agit donc d'un problème plus simple que celui vu en TP. On veut écrire une procédure qui fusionne ces deux monotonies et écrit la monotonie résultante dans un troisième fichier. Cette procédure prend comme *seuls* paramètres les noms des deux fichiers d'entrée (**nomfic1** et **nomfic2**) et le nom du fichier de sortie (**nomficSortie**).

Préconditions :

- nomfic1 et nomfic2 sont des fichiers binaires qui contiennent chacun une séquence triée (monotonie) de nombres au format IEEE 754 double précision. Si l'un de ces fichiers ne peut pas être ouvert (par exemple parce qu'il n'existe pas), alors le programme se termine avec un code d'échec.
- Les deux séquences peuvent être de longueur différente, mais elles sont au moins de longueur 1 : autrement dit, chaque fichier contient au moins un élément.

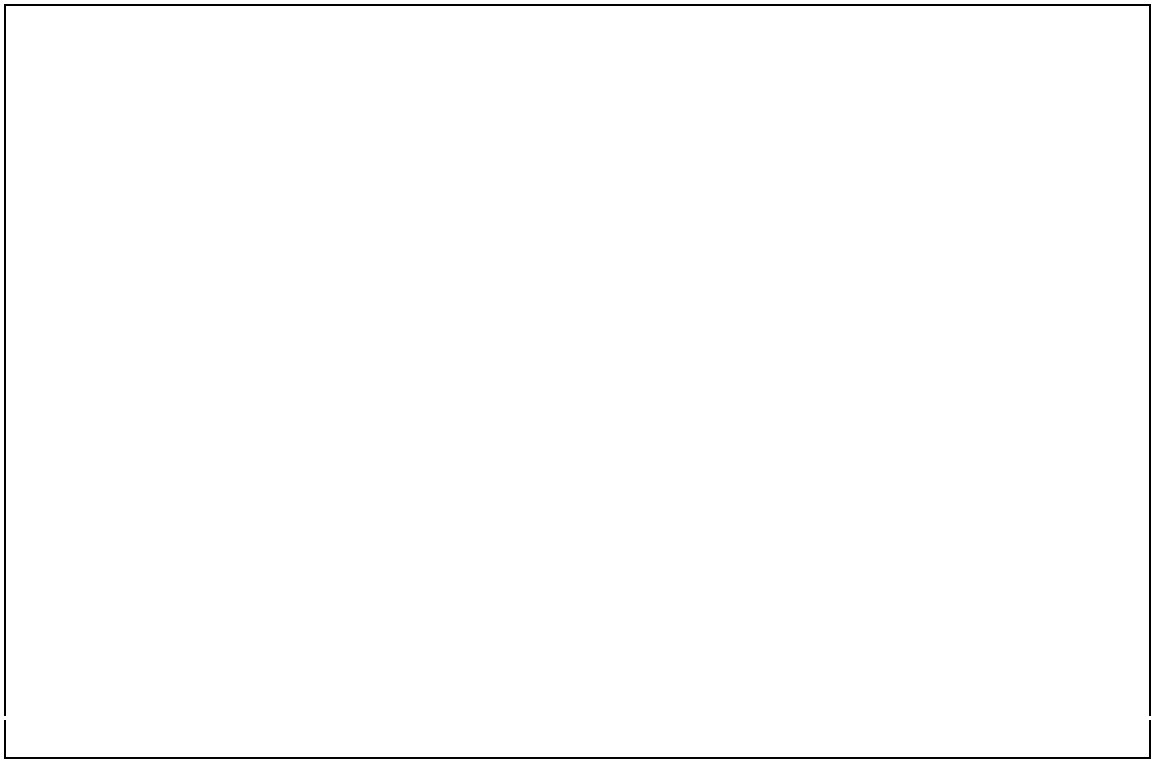
Postconditions :

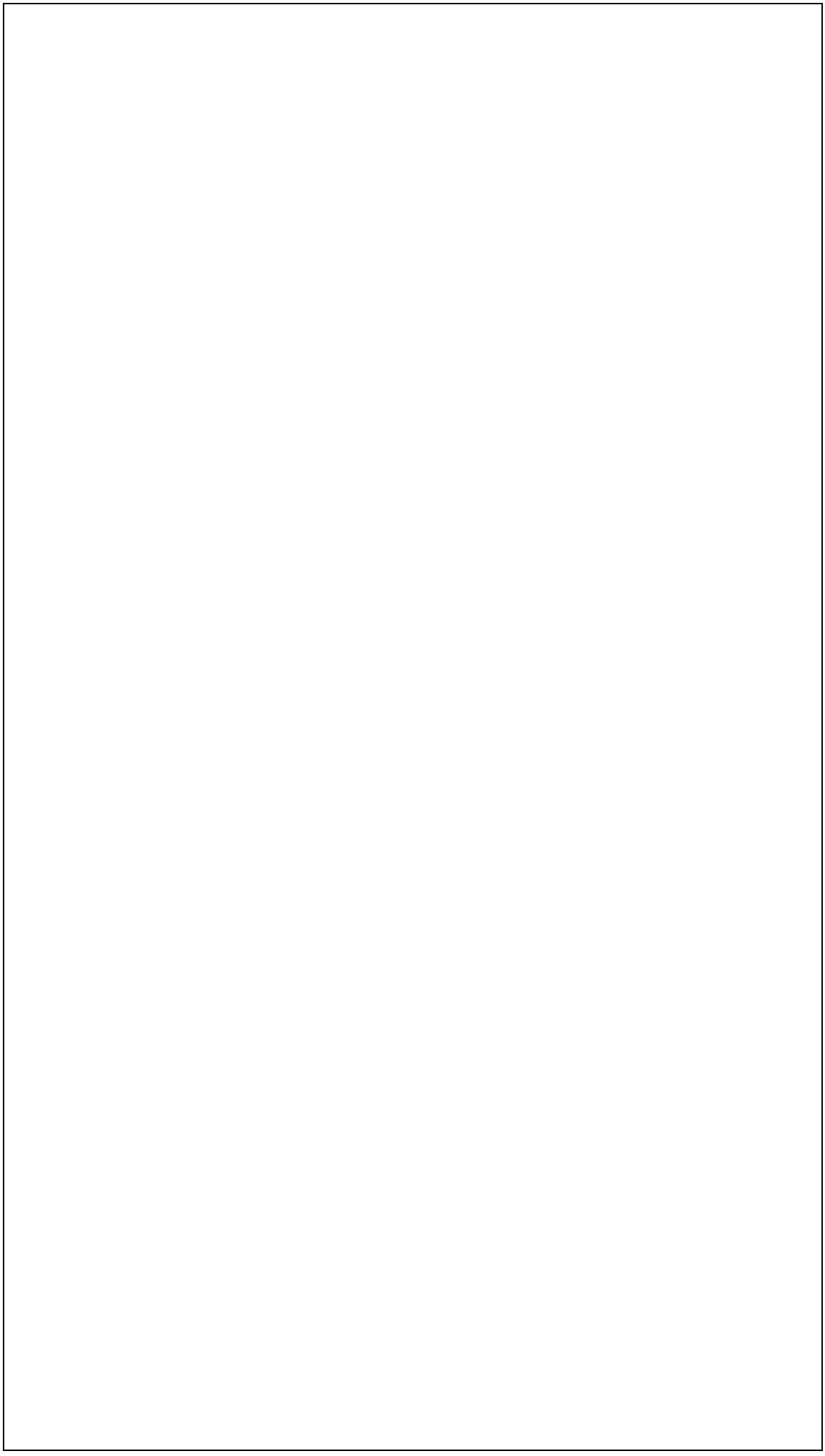
- Un nouveau fichier binaire nommé nomficSortie est créé (s'il existait déjà, son contenu est écrasé). Ce fichier contient une monotonie correspondant à la fusion des deux monotonies contenues dans nomfic1 et nomfic2. Si ce fichier ne peut pas être créé (par exemple à cause d'un problème de droits insuffisants dans le répertoire courant), le programme se termine avec un code d'échec.
- Les contenus des fichiers nomfic1 et nomfic2 sont inchangés.

Donnez le code C de cette procédure. Vous utiliserez les variables locales suivantes (à vous d'en préciser le type) : **e1, e2, fic1, fic2, ficSortie, succeslect1, succeslect2**. Vous pouvez en ajouter d'autres si nécessaire.

On rappelle l'ordre des paramètres pour les sous-programmes suivants, à vous de déterminer lequel ou lesquels doivent être utilisés ici :

- fscanf(pointeur sur le flot, chaîne de format, adresse variable 1, adresse variable 2...)
- fprintf(pointeur sur le flot, chaîne de format, valeur variable 1, valeur variable 2...)
- fread(adresse 1er bloc, taille d'un bloc, nombre de blocs, pointeur sur le flot)
- fwrite(adresse 1er bloc, taille d'un bloc, nombre de blocs, pointeur sur le flot)





Exercice 3 : Trace mémoire d'un programme (8 points)

Considérons le programme C suivant :

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

void mystere(unsigned int i, unsigned int nbr, char t[])
{
  unsigned int reste;
  if (nbr != 0)
    {
      reste = nbr % 16;
      if (reste <= 9) {t[i] = reste + '0';}
      else {t[i] = reste - 10 + 'a';}
      mystere(i-1, nbr / 16, t);
    }
  else
    {
      /*Dessin 1= etat de la memoire quand on rentre dans ce else*/
    }
}


int main()
{
  char monTab[9];
  char * c = 0x0;
  unsigned int k;
  unsigned int monNombre = 455;

  for(k = 0; k < 8; k++) {monTab[k] = '0';}
  monTab[k] = '\0';

  mystere(7, monNombre, monTab);
  printf("%s\n", monTab);

  c = (char *) malloc(9*sizeof(char));
  for(k = 0; k < 9; k++) {*(c + k) = monTab[k];}
  /* Dessin 2 = etat de la memoire a ce moment */

  free(c);

  return 0;
}
```

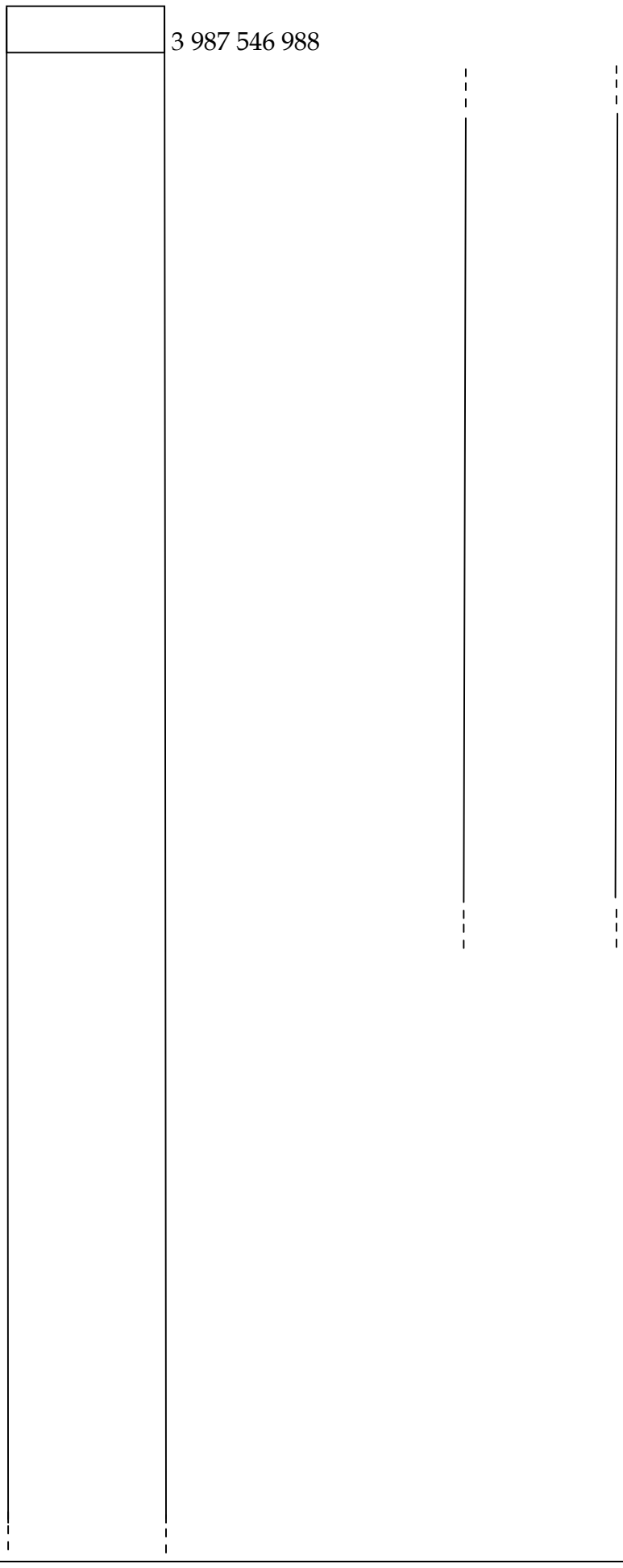
1. Dessinez l'état de la pile et du tas aux deux moments indiqués en commentaires (on vous demande pour chaque dessin une « photo » de la pile entière et de tout ce qu'il y a dans le tas). Vous utiliserez le modèle théorique de pile vu en cours et en TD. Vous supposerez que la valeur de retour du main est stockée à l'adresse 3 987 546 988 (voir cadre-réponse page suivante).

   A toutes fins utiles :
   16 x 1 = 16
   16 x 2 = 32
   16 x 3 = 48
   16 x 4 = 64
   16 x 5 = 80
   16 x 6 = 96
   16 x 7 = 112
   16 x 8 = 128
   16 x 9 = 144
   16 x 10 = 160

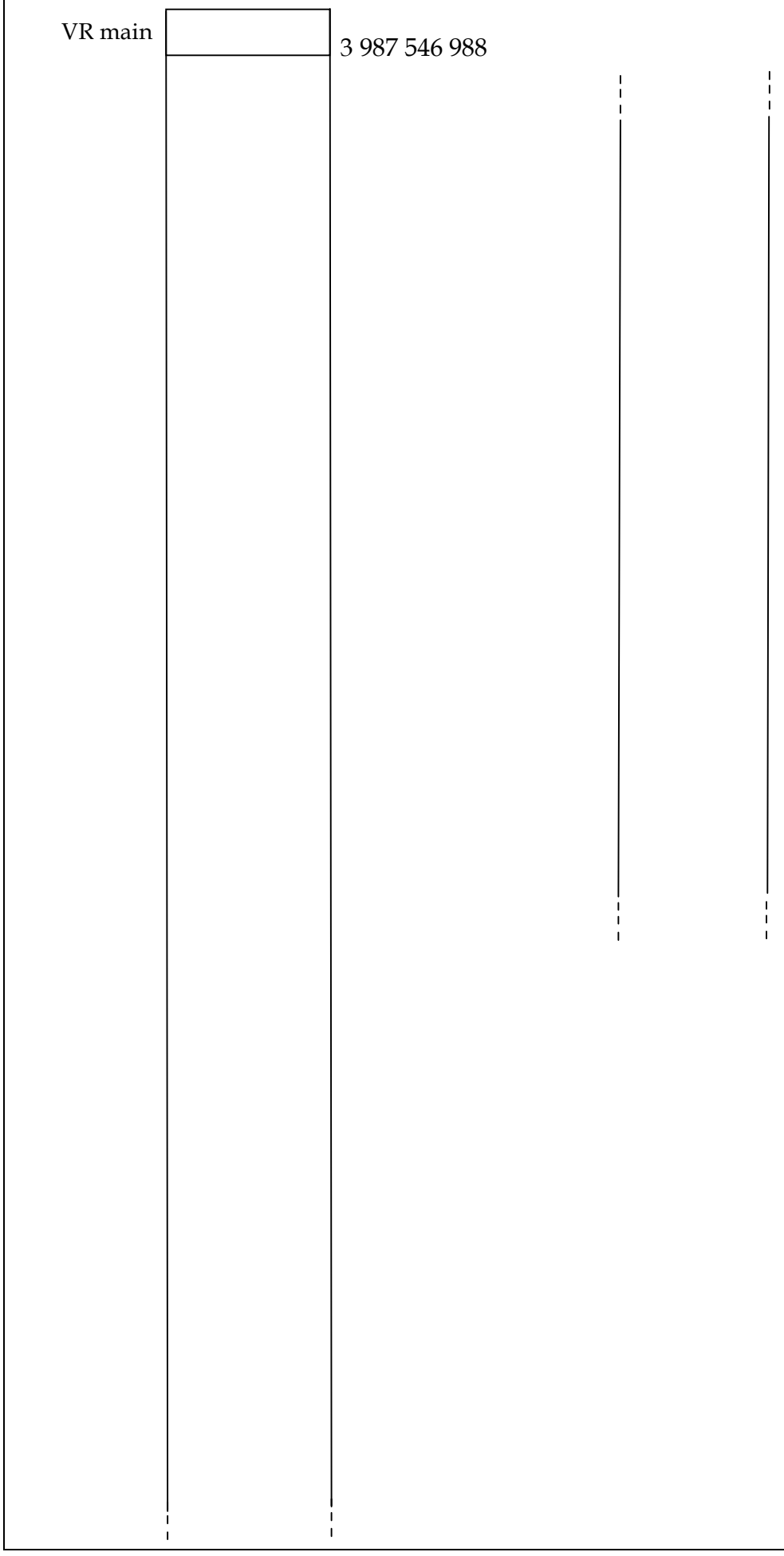
Dessin 1 : **PILE** **TAS**

VR main

3 987 546 988



Dessin 2 : **PILE** **TAS**



2. Indiquez ce qu'affiche le printf du main lorsqu'on exécute ce programme précis.

3. Que fait la procédure mystere ? Autrement dit, si vous deviez lui donner un nom plus explicite, lequel choisiriez-vous ?

4. Si les unsigned int sont codés sur 32 bits, risque-t-on un « buffer overflow » lorsqu'on remplit monTab (tableau de 9 char) avec la procédure mystère ? Justifiez votre réponse.