Le tableau ci-dessous représente une mémoire d'ordinateur.

Chaque case est repérée par un numéro (sur fond jaune).

Les cases contiennent des informations de diverses natures : entiers, réels, enregistrements de type `t_cellule`, etc.

Des messages y sont inscrits. Il s'agit de les retrouver en suivant un jeu de pistes.

Exemple :

Un message commence en 87.
La case 87 contient le caractère 'O' et le numéro de la prochaine case : D4.
La case D4 contient le caractère 'u' et le numéro de la prochaine case : F2.
La case F2 contient le caractère 'i' et null qui indique le message est terminé.
Le message commençant en 87 est donc la suite de caractères 'O', 'u', 'i' (le mot "Oui").

Question 1.a Quel est le message qui commence en C7 ?
Question 1.b Quel est le message qui commence en 09 ?
Question 1.c Quel est le message qui commence en 21 ?

00	10 'h', 22	20 'D', 51	30 '2', 01	40 'B', 31	50 'c', 51	60 'J', 98	70 'y', 70	80 'c', 51	90 '.', null	A0 '7', 32	B0 'j', 74	C0 'n', 53	D0 'e', A1	E0 'g', 23	F0 'j', 81
01 'n', 5B	11 "bol"	21 'u', 01	31 'é', 08	41 'e', 43	51 '´', C6	61 'e', 28	71 'i'	81 'c', 51	91 'c', 51	A1 'a', B6	B1 'c', 51	C1 't', 31	D1 'j', F5	E1 'e', 69	F1 "ut"
02 'f', 12	12 5	22 'u', C9	32 '*', 20	42 'b', AA	52 'c', 51	62 'n', 56	72 'c', 51	82 "pas"	92 'e', 90	A2 'y', 54	B2 'c', 51	C2 'x', 72	D2 '/', 31	E2 'i', C8	F2 'i', null
03 'e', 14	13 'c', 51	23 'e', B5	33 'u', 58	43 ' ', 86	53 'é', 92	63 'a', C5	73 'c', 51	83 'c', 51	93 'c', 51	A3 'm', 27	B3 11	C3 "œil"	D3 'j', 1A	E3 "yeux"	F3 'l', F8
04 52	14 ' ', 15	24 'p', A2	34 'A, null	44 'd', 17	54 'c', 51	64 "si"	74 'p'	84 ' ', 2B	94 'c', 51	A4 'c', 21	B4 "brr"	C4 'c', 51	D4 'u', F2	E4 'o', 87	F4 'k', BA
05 'a', 28	15 'l', 55	25 2,2	35 'U', 6B	45 21	55 'i', 6C	65 'h', A3	75 'c', 51	85 'c', 51	95 'i', 9B	A5 "do"	B5 'c', 51	C5 'î', C0	D5 '4', 51	E5 'r', 41	F5 "fa"
06 3,21	16 'h'	26 'k', 57	36 't', null	46 '0', 56	56 'c', 51	66 'e, 77	76 'c', 51	86 'l', 95	96 "sol"	A6 'q', 47	B6 'o', B1	C6 'e', FC	D6 'm', 51	E6 'l', 94	F6 "Paris"
07 'g', 1C	17 'z'	27 'r', 66	37 'k, 4F	47 '+', 94	57 'u'	67 'é', 36	77 'a', F3	87 'O', D4	97 'n', 44	A7 "or"	B7 'u', ED	C7 's', 77	D7 'm', 51	E7 'i', 6C	F7 'c', 51
08 31	18 'z', AA	28 'Q', 37	38 'p', 45	48 'V', F0	58 555	68 'ç', 39	78 'c', 51	88 ' ', 79	98 2020	A8 "ré"	B8 'c', 51	C8 "main"	D8 '$, 51	E8 'a', 7B	F8 'u', 36
09 'C', 51	19 "ma"	29 'Y', 05	39 'n', 03	49 "sans"	59 'h', 63	69 'r', B5	79 'a', B7	89 'c', 51	99 1	A9 'e', BA	B9 'u, A1	C9 'h', AF	D9 't', FD	E9 'e', AA	F9 "fou"
0A 'r', 52	1A "ta"	2A 'I', 20	3A 'v', 5A	4A 'c', 81	5A 'b', 23	6A 'n', A0	7A 'c', 51	8A 'c', 51	9A 22	AA 'c', 51	BA ' ', 1E	CA 'd', DD	DA 'c', 51	EA 'o', 48	FA "bla"
0B 'a', 51	1B "sa"	2B 'u', 39	3B 'W', 57	4B 'è', 5A	5B 'e', 88	6B 't', A9	7B 'h', null	8B 'c', 51	9B 's', D9	AB 'f', 3A	BB '*', 65	CB 't', 51	DB 'z, 44	EB 'm', BA	FB 'p', 12
0C 'e', 52	1C "ton"	2C 'j', 31	3C 'a', B5	4C 'y', 8F	5C 'r', 87	6C 's', 6B	7C '>', 55	8C 'c', 51	9C 236	AC 'm', B1	BC 's', 55	CC "hiii"	DC 'h', 36	EC 'w', BB	FC 's', 7D
0D 'i', 53	1D "tes"	2D 'q', 58	3D 'z', 42	4D 'h', 5C	5D '7', 2C	6D '-', 57	7D 't', 84	8D 'c', 51	9D 2	AD 'ù', 3F	BD 'v', 18	CD 'r', 22	DD 'c', 53	ED 't', E5	FD 'e', null
0E 'o', 54	1E 'c', 59	2E 't', 54	3E '4', 10	4E 'a', 53	5E 'r', 5F	6E 'c', 58	7E 'y', 9C	8E 'c', 51	9E 'y', 78	AE "CO2"	BE 'y', 57	CE '7', 31	DE '8', 21	EE "aux"	FE 'a', 52
0F 'u', 55	1F 'o', 2E	2F 'z', 62	3F '.', B0	4F 's', 32	5F 'é', 88	6F '%', 7F	7F '>', 41	8F 'c', 51	9F 'n', 84	AF 's', 27	BF '?', 7A	CF '!', 61	DF "grrr"	EF "ail"	FF 'p', AA

Implémentation

Un chaînage est donc composée de cellules qui sont chaînées les unes aux autres en suivant les adresses mémoires

Question 2 : implémenter les types et algorithmes suivants, et faire fonctionner le programme.

```
type t_cellule = enregistrement
       Caractère info
       pointeur vers t_cellule suivant
fin enregistrement

//-----------------------------------------------
rôle : enregistrement en mémoire d'un message saisi par
l'utilisateur. La fin du message est marquée par un point. La
fonction retourne l'adresse de début du chainage

fonction enregistrement() : pointeur vers t_cellule
Caractère lettre
pointeur vers t_cellule pcel, psuiv, tete
début
       ecrire "une lettre ? (un point pour terminer)"
       lire lettre
       pcel ← allocation t_cellule
       // ..... début de la liste
       tete ←  pcel

       tant que (lettre != '.') faire
              (mémoire pcel).info ← lettre
              psuiv ← allocation t_cellule
              (mémoire pcel).suivant ← psuiv
              ecrire "une lettre ? (un point pour terminer)"
              lire lettre
              pcel  ←  psuiv
       fin tant que
       // ..... mémorisation du point final
       (mémoire pcel).info ← lettre
       // ..... terminaison de la liste
       (mémoire pcel).suivant ← null

       retourner tete
fin
```

```
//-----------------------------------------------
rôle : affiche un message qui commence à l'adresse pdeb
procédure affiche(d pointeur vers t_cellule pdeb)
Caractère lettre
pointeur vers t_cellule pcel
début
       pcel ← pdeb
       tant que (pcel != null) faire
              lettre ← (mémoire pcel).info
              ecrire lettre
              pcel ← (mémoire pcel).suivant
       fin tant que
       ecrire endl
fin

//-----------------------------------------------
// programme principal
//
pointeur vers t_cellule pcel
début
       pcel ←  enregistrement()
       affiche(pcel)
fin
```

Question 3.a :
L'utilisateur enregistre le message `'s', 'a', 'l', 'u', 't', '.'`.
A la fin de l'exécution, combien reste-t-il de cases mémoires allouées dynamiquement par les programmes et sous-programmes ?

Question 3.b :
Créer un sous-algorithme qui désalloue toutes les cases mémoires qui ont été allouées dynamiquement.
Ce sous-algorithme sera appelé par le programme principal, juste après l'affichage du message et avant de terminer.

Question 4 :
Ecrire un sous-algorithme qui crée un chaînage à partir d'une chaîne de caractères.

Manipulations

L'emploi du temps d'un étudiant à l'université comporte plusieurs séances pour chaque Unité d'Enseignement, occupant chacune un créneau. Le but de cet exercice est de proposer des outils permettant de gérer les séances d'une UE d'une manière souple, des séances pouvant être déplacées, supprimées ou ajoutées au fur et à mesure.

Voici une proposition de définition de type pour représenter une succession de séances, les créneaux étant supposés tous durer 1h20.

```
type t_seance = enregistrement
       Entier  numeroSemaine
       Entier jourSemaine // 1=lundi, 2=mardi, ...
       Entier heures // de 8 à 18
       Entier minutes // 0 ou 30
       pointeur vers t_seance    seanceSuivante
fin enregistrement
```

Question 5.a : Définir un sous-algorithme prenant en paramètre un pointeur vers une première t_seance et donnant le nombre total de séances prévues dans le chaînage qui suit. Le pointeur null passé en paramètre désigne une absence de séance et la dernière séance d'un chaînage non vide a son champ seanceSuivante à null.

Question 5.b : Définir un sous-algorithme prenant en paramètre un pointeur vers un t_seance et affichant pour chaque séance du chaînage sur une ligne le numéro de la semaine, le jour et l'heure (par exemple `Semaine 48 lundi 15h30`).

Question 5.b : Définir un sous-algorithme prenant en paramètre un pointeur vers un t_seance et désalloue tout le chaînage commençant avec ce pointeur. Le pointeur null doit être accepté en entrée et dans ce cas le sous-algorithme ne fait rien.

Question 6.a : Définir un sous-algorithme prenant en paramètre un pointeur vers un t_seance et les informations d'un créneau (semaine, jour, heures et minutes) et ajoutant une séance au chaînage donné.

Question 6.b : Définir un sous-algorithme prenant en paramètre un pointeur vers un t_seance et les informations d'un créneau (semaine, jour, heures et minutes) et supprimant la séance correspondante du chaînage donné. Si le créneau n'y figurait pas, alors l'information doit en être fournie comme résultat et le chaînage ne doit pas être modifié.

Question 6.b : Définir un sous-algorithme prenant en paramètre un pointeur vers un t_seance et les informations d'un créneau (semaine, jour, heures et minutes) et donnant le nombre de séances prévues (restantes) après (strictement) ce créneau.

Question 7.a : Refaire les questions 6 en supposant que le chaînage est maintenu en permanence trié (et le maintenir trié dans votre programmation).

Question 7.b : Discuter de l'intérêt de maintenir le chaînage trié en termes de complexité temporelle et spatiale.

Question 8 : Définir un algorithme de fusion de deux chaînages de séances, avec suppression des doublons (un même créneau — même semaine, même jour, même heure et même minutes) ne doit pas y figurer deux fois). Proposer une première version qui construit un nouveau chaînage sans détruire les deux autres (il faudra dupliquer les informations), puis une seconde qui modifie sur place le premier chaînage en détruisant le second, mais en ne créant aucun t_seance et en désallouant éventuellement les superflus (en cas de doublons).