

Université Côte d'Azur Programmation C

PO2 SITE

EXAMEN: PROGRAMMATION C 11 JANVIER 2023

Durée: 1 heure

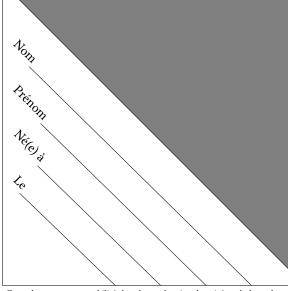
Tous documents autorisés. Il est interdit d'accéder à internet.

Note

Toutes les questions sont indépendantes. Tous les codes devront être écrits en Langage C.

La notation est donnée à titre indicatif.

Nombre de pages : 6



Il est de votre responsabilité de rabattre le triangle grisé et de le cacheter au moyen de colle, agrafes ou papier adhésif. Si ne vous le faites pas, vous acceptez implicitement que votre copie ne soit pas anonyme.

Exercices divers (7 points)

1. Écrire une caractères con	fonction n	ombre_c t à des ch	hiffre niffres. I	es qui pi Par exen	end en a aple nom	argumen bre_chi	t une ch	aîne de ("abc 4	caractè 45+") r	res et re enverra	nvoie l 2.	e nom	bre de
initialise t3 e	en affectant	chaque ·	t3[i] (n) à la p	lus gran		valeurs	entre t1	[i] et	t2[i].		enant,
2. Écrire une initialise t3 e par exemple,	en affectant	chaque ·	t3[i] ((avec i<	n) à la p	lus gran	de des v	valeurs	entre t1	[i] et	t2[i].	En pro	enant,
initialise t3 e	en affectant	chaque ·	t3[i] ((avec i<	n) à la p	lus gran	de des v	valeurs	entre t1	[i] et	t2[i].	En pro	enant,
initialise t3 e	en affectant	chaque ·	t3[i] ((avec i<	n) à la p	lus gran	de des v	valeurs of devra	entre t1	[i] et ialisé à	t2[i].	En pro	enant,
initialise t3 e	en affectant	chaque ·	t3[i] ((avec i<	n) à la p	lus gran	de des v	valeurs of devra	entre t1 être init	[i] et ialisé à	t2[i].	En pro	enant,
initialise t3 e	en affectant	chaque ·	t3[i] ((avec i<	n) à la p	lus gran	de des v	valeurs of devra	entre t1 être init	[i] et ialisé à	t2[i].	En pro	enant,
initialise t3 e	en affectant	chaque ·	t3[i] ((avec i<	n) à la p	lus gran	de des v	valeurs of devra	entre t1 être init	[i] et ialisé à	t2[i].	En pro	enant,
initialise t3 e	en affectant	chaque ·	t3[i] ((avec i<	n) à la p	lus gran	de des v	valeurs of devra	entre t1 être init	[i] et ialisé à	t2[i].	En pro	enant,
initialise t3 e	en affectant	chaque ·	t3[i] ((avec i<	n) à la p	lus gran	de des v	valeurs of devra	entre t1 être init	[i] et ialisé à	t2[i].	En pro	enant,
initialise t3 e	en affectant	chaque ·	t3[i] ((avec i<	n) à la p	lus gran	de des v	valeurs of devra	entre t1 être init	[i] et ialisé à	t2[i].	En pro	enant,
initialise t3 e	en affectant	chaque ·	t3[i] ((avec i<	n) à la p	lus gran	de des v	valeurs of devra	entre t1 être init	[i] et ialisé à	t2[i].	En pro	enant,

3. Que fait le code suivant? On ne vous demande pas de paraphraser. Vous décrirez sa fonction et vous l'illustrerez

```
par un exemple.
int mystere(char * c) {
   char * oo = c ;
   while (*(c++) != ' \setminus 0');
   return c - oo - 1 ;
4. Écrire une fonction echange, qui prend en argument deux pointeurs vers des flottants et qui échange leurs va-
leurs. Par exemple, si a pointe vers la valeur 3.5 et b pointe vers la valeur 2.0, l'appel de echange (a,b) fera pointer
a vers 2.0 et b vers 3.5.
```

5. Utiliser la fonction echange pour compléter le code suivant. À la fin, les valeurs de x et y doivent être permutées. Votre code ne doit pas dépasser une ligne.

```
float x = \overline{2.0};
float y = 3.0;
/* ligne à compléter pour que x vaille 3.0 et y 2.0 */
```

Programmation C 3/6

Problème (13 points)

On cherche dans cet exercice à implémenter une file à partir de deux tableaux. Ce n'est pas forcément optimal, mais c'est un prétexte pour faire du C. En pratique, la file est séparée en deux tableaux : un premier auquel on ajoute les éléments et un second, duquel on les retire. Lorsque le second tableau est vide, on transfère le contenu du premier vers celui-ci.

Exemple. Prenons la file abstraite suivante : \rightarrow 3 5 7 2 1 \rightarrow ; elle sera impléméntée en C par les deux tableaux ci-dessous.

```
• entree : \boxed{7} \boxed{5} \boxed{3} \boxed{\phantom{0}} \leftarrow (notez comme l'ordre est inversée par rapport à la file) 
• sortie : \boxed{2} \boxed{1} \boxed{\phantom{0}} \rightarrow
```

Les cases vides, représentent des valeurs indéfinies. Pour connaître à chaque instant le nombre de cases effectivement remplies on ajoute deux variables correspondant à des indices.

- indice_entree represente la dernière case remplie du tableau entree (ici, la troisième case d'indice 2).
- $-\,$ indice_sortie represente la première case vide du tableau sortie (ici aussi, la troisième case d'indice 2).

Ainsi, si j'ajoute l'élément 10 (à la fin de la file) et que je supprime l'élément du début de la file (ici, 1), j'obtiens comme nouvelle file $\xrightarrow{}$ 10 | 3 | 5 | 7 | 2 | $\xrightarrow{}$ La file sera alors codée par :

```
    entree: 7 5 3 10  ← (et indice_entree vaut maintenant 3)
    sortie: 2  → (et indice_sortie vaut maintenant 1)
```

```
On vous donne la structure suivante définissant le type file.
 typedef struct {
   int entree[N] ;
   int indice_entree ;
   int sortie[N] ;
   int indice sortie ;
} file ;
1. Définir la valeur N à 10 à l'aide d'une macro (en utilisant le préprocesseur).
2. Que faut-il écrire comme code pour créer une nouvelle file sur la pile avec les indices correctement initialisés.
3. Écrire une fonction est_vide qui prend une file en paramètre et renvoie un booléean (c'est-à-dire un entier) pour
indiquer si la file est vide.
```

Programmation C 4/6

														mètre (et non un pointeur) et un entier x et renvoie une entree est déjà complet, la fonction renverra NULL.
														ment un pointeur vers une file (on supposera le tableau e procédure modifiera la file accessible via le pointeur.
•				• • •				• • • •				• • • •		
								• • • •				• • • •		
								• • • •				• • • •		
				• • •				• • • •				• • • •		
														aleur et que le tableau sortie est vide. Écrire la fonction s le tableau sortie. La file sera passée en argument sous
														r (si le tableau sortie était non vide) et 1 sinon.
	•	12		15	1	3	10							: entree
	•													: sortie
En	pa	rtan	ıt o	de l'	éta	t p	récé	dent (et en a	appela	nt la	proc	édure	transfert, on obtient l'état ci-dessous.
	•													: entree
	•	10		13	1	5	12							: sortie
•			•											
•														
•			•••	• • •				• • • •						
•			• •					• • • •				• • • •		
•			• •	• • •	• • •			• • • •			• • • • •	• • • •		

Programmation C 5/6

7. En utilisant la question précédente, réécrivez la fonction retrait, mais en commençant par transférer les donnée si la liste sortie est vide. La fonction prendra en argument un pointeur vers une file et renverra -1 si la file est vide.
On modifie maintenant la structure file afin de pouvoir changer dynamiquement la taille des tableaux.
typedef struct {
<pre>int n; /* taille des tableaux entree et sortie */ int * entree ; int indice_entree ; int * sortie ; int indice_sorties ; } file ;</pre>
8. Écrire une fonction <u>initialisation</u> qui prend en argument un entier <i>n</i> et qui renvoie un pointeur vers une fil correctement allouée sur le tas.
correctement allouée sur le tas.
correctement allouée sur le tas.
correctement allouée sur le tas.
correctement allouée sur le tas.
correctement allouée sur le tas.
correctement allouée sur le tas.
correctement allouée sur le tas.
correctement allouée sur le tas.
correctement allouée sur le tas.
correctement allouée sur le tas.
correctement allouée sur le tas.
9. Écrire un fonction liberer qui prend en argument un pointeur vers une file et qui désalloue proprement la mé
9. Écrire un fonction liberer qui prend en argument un pointeur vers une file et qui désalloue proprement la mé
9. Écrire un fonction liberer qui prend en argument un pointeur vers une file et qui désalloue proprement la mé
9. Écrire un fonction liberer qui prend en argument un pointeur vers une file et qui désalloue proprement la mé

Programmation C 6/6

10. Réecrire la fonction ajout pour doubler la valeur de n si le tableau entree est complet.