Dauphine | PSL
UNIVERSITÉ PARIS

MIDO – L2 MIE – 2022-2023

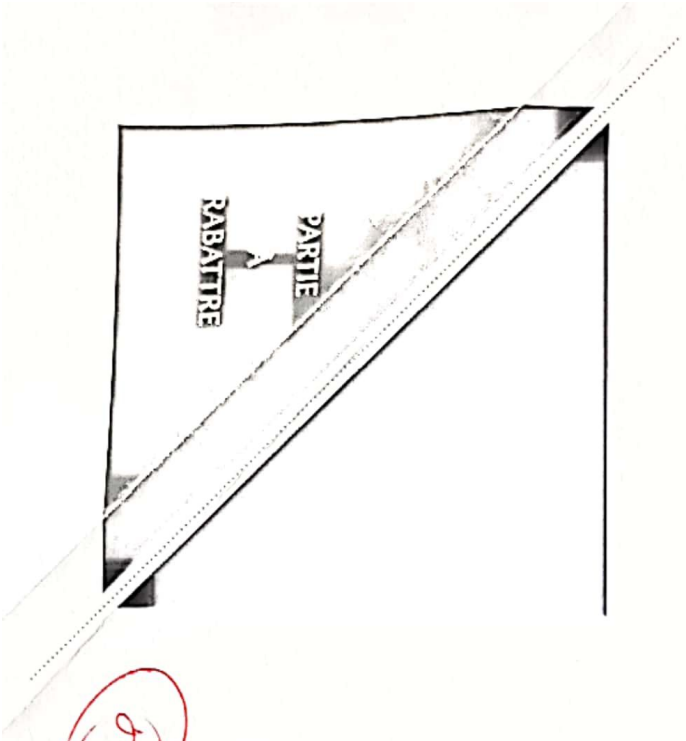
Programmation C

Examen du 30 juin 2023 (2e session)
(durée 2h)

Répondez directement sur le sujet.

Documents autorisés : une feuille A4 manuscrite recto-verso
Calculatrice autorisée

9,5



Exercice 1

Cochez la ou les bonne(s) réponse(s)

1. Avec l'instruction :

```
short *p = malloc(sizeof(short) * 6);
```

- ☐ je ne peux utiliser p que pour stocker un entier ;
- ☒ je peux utiliser p comme un identificateur de tableau de 6 entiers courts ;
- ☐ je peux utiliser p pour stocker autant d'entiers courts que je veux ;
- ☐ la déclaration est illégale en C.

2. L'instruction :

```
char i, j = 0;
```

- ☐ déclare deux variables entières, toutes les deux initialisées à zéro ;
- ☒ déclare deux variables entières, une non-initialisée, l'autre initialisée à zéro ;
- ☐ déclare une variable entière et une autre nulle et sans type ;
- ☐ la déclaration est illégale en C.

3. L'expression :

```
a ? 1 : 0
```

- ☐ renvoie 1 si a vaut 0 ou 1 ;
- ☐ renvoie 1 si a vaut 1 et 0 sinon ;
- ☐ renvoie 0 si a est non-nul et 1 sinon ;
- ☒ renvoie 0 si a est nul et 1 sinon ;
- ☐ est équivalent à l'expression a ;
- ☐ n'a pas de sens en C.

4. L'instruction :

```
int *pi = &i;
```

- ☐ définit un pointeur `pi` toujours initialisé à `NULL` ;
- ☐ définit un pointeur `pi` non initialisé ;
- ☒ définit un pointeur `pi` pointant sur une variable `i` existante ;
- ☐ définit un entier `pi`.

Exercice 2

Soit le programme suivant :

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

int main() {
    char str[] = "pcpc";
    char *p1 = str;
    char *p2 = str + strlen(str);

    while ((*p1)++) {
        *(--p2) = (*p1)++;
        p1++;
    }
    *p1 = '\0';
    printf("%s\n", str);
    return 0;
}
```

Remarque : la fonction `strlen(str)` renvoie la longueur de la chaîne passée en paramètre (c'est-à-dire le nombre de caractères avant le premier `'\0'` rencontré).

1. Ce programme compile-t-il et s'exécute-t-il sans erreurs ?

Non, ce programme ne compile pas sans erreur.

2. Si non, pourquoi ?
   Si oui, qu'est-ce que l'exécution du programme affiche-t-elle ?

La commande while ((*p1)++) { incrémente le caractère et non pas la position du pointeur ce qui crée une erreur au moment de compiler.

9,5

Exercice 3

Voici une fonction.

Son rôle est, à partir d'une chaîne de caractère passée en argument, de renvoyer une nouvelle chaîne composée uniquement des chiffres présents dans la chaîne initiale. Ainsi, si la fonction est appelée avec l'argument "10 inventent 1, 2, 3, soleil !", la fonction est supposée renvoyer "10123".

Malheureusement, bien que cette fonction compile sans erreurs ni avertissements, le programmeur y a laissé plusieurs bugs. Saurez-vous les trouver tous ?

Listing 1. Une fonction erronée

```
1  char *extract(char *str) {
2      char *pdigits = malloc(strlen(str));
3      int i = 0;
4
5      do {
6          if (str[i]>'0' && str[i]<='9')
7              *pdigits++ = str[i];
8          if ((str[++i] = '\0'))
9              break;
10     } while (0);
11     return pdigits;
12 }
```

ligne 2: char *pdigits = malloc(strlen(str)+1); // si jamais str contient // déjà que des chiffres, il faut une place pour '\0', ✓

ligne 6: if (str[i] >= '0' && str[i] <= '9') // sinon cela reprend pas 0. ✓

ligne 7,8: dans le premier if on rajoute entre la ligne 7 et 8 : i++; ← et dans le if ?

ligne 8: if ((str[++i] == '\0')) // test d'egalité "==" et pas "=". ✓

ligne 10: } while (1); ✓

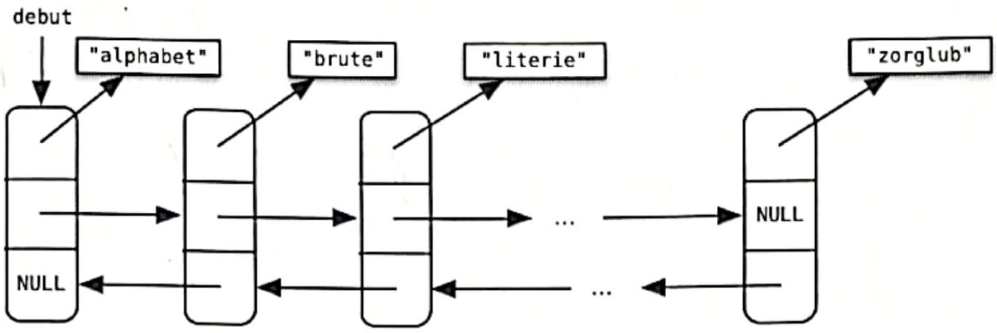
ligne 11: return *pdigits; deuxième ✓

ligne 9-9: après le if on rajoute : *pdigits++ = '\0'; // afin de terminer notre chaine. ✓

5

Exercice 4

On souhaite gérer une liste doublement chaînée de chaînes de caractères de telle sorte que la liste soit toujours triée (la variable globale `debut` pointe vers la tête de la liste).



On pourra utiliser les fonctions suivantes :

- `int strcmp(char *s1, char *s2);` permet de comparer des chaînes de caractères et renvoie 1 si `s1 > s2`, 0 si `s1 = s2` et −1 si `s1 < s2`;
- `char *strdup(char *str);` permet de dupliquer une chaîne de caractères passée en argument en allouant la place mémoire nécessaire.

Dans toutes les fonctions suivantes, la liste peut être vide (et dans ce cas, le pointeur de tête `debut` vaut `NULL`). **Il faudra également lorsque cela est nécessaire mettre à jour le pointeur global `debut`.**

1. Définir la structure correspondante

   `struct cellule`

   ainsi que le pointeur global `debut`.

typedef struct cellule {
    char *ch;
    struct cellule *next;
    struct cellule *prev;
} Cellule;

Cellule *(debut = NULL)

1

2. Écrire une fonction

```
int compte(void);
```

qui compte le nombre d'éléments de la liste.

```
int compte (void) {
    int cpt = 0;
    if (debut == NULL) {
        return cpt;
    } else {
        while (debut) {
            cpt ++;
            debut = debut -> next;
    }
    return cpt;
}
```

2

3. Écrire une fonction

```
void insere(char *str);
```

qui insère une nouvelle chaîne au bon endroit dans la liste.

```
void insere(char *str){
    Cellule courant = debut;
    Cellule *q = strdup(str);
    q -> ch = str;
    q -> next = NULL;
    if (debut == NULL){
        printf("la liste est vide");
    }
    else if (strcmp(str, (debut -> ch)) < 0){
        q -> next = debut;
    }else{
        while (courant -> next && strcmp((courant -> next -> ch), str) < 0){
            courant = courant -> next;
        }
        q -> next = courant -> next;
        courant -> next = q;
    }
}
```

malloc

strdup ici

Ajoutez la cellule

+ modif debut ?

1,5

q -> pred ?

q -> next -> pred ?

4. Écrire une fonction

```
void supprime(char *str);
```

qui supprime de la liste une chaîne passée en argument si elle s'y trouve.

```
void supprime (char *str) {
    Cellule *p;
    while (debut && strcmp (debut->ch, str) != 0) {
        debut = debut -> next;
    }
    if (strcmp (*(debut -> next -> ch), '\0') == 0) {
        printf ("La chaine passé en argument n'est pas dans
                la liste");
    } else {
        p = debut -> next;
        free (debut);
        debut = p;
    }
```

Si chaîne pas trouvée debut = null

// on est arrivé au bout de la liste

Vous supprimez tout le début de la liste, pas uniquement la cellule