+1/1/60+

QCM **C/SHELL**

Langage C et Shell
Examen du 10/03/2020

Nom et prénom :

.....

Durée : 1h30min.
Une feuille A4 recto-verso de notes est autorisée. L'usage de la calculatrice ou d'un ordinateur est interdit.
Les questions faisant apparaître le symbole ♣ peuvent présenter une ou plusieurs bonnes réponses. Les autres ont une unique bonne réponse.
*Les cases sont à **cocher** d'une croix et **NON à griser** ! Une case grisée est équivalente à une case vide (cela vous permet de corriger une erreur sans essayer d'effacer une croix, il suffit de griser entièrement la case).*
1 point par bonne réponse, -0.5 point par mauvaise réponse. 0 point si aucune case n'est cochée. Une réponse incohérente (par ex. plusieurs cases cochées alors que la question n'attendait qu'une seule réponse) entraîne une note de -0.5 à la question correspondante. Une question avec le symbole ♣ est sur 3 points (cela ne veut pas dire qu'il y a nécessairement 3 bonnes réponses), avec une note plancher à -1. La question ouverte 3 est notée sur 4 points. Les questions ouvertes 11 et 12 sont notées sur 2 points. Les notes des questions ouvertes ne peuvent pas être négatives. La note finale ne peut pas être négative.

Question 1 Le code suivant est compilé avec la commande `gcc main.c` et on l'exécute, si possible, avec `./a.out`. Que se passe-t-il ?

```
#include <stdio.h>

int main() {
    int *ptr_i = 1;
    int i = *ptr_i;
    printf("%d\n", i);
    return 0;
}
```

- ☐ Affiche `1`
- ☐ Affiche `0`
- ☐ Erreur de segmentation
- ☐ Ça ne compile pas

Question 2 Quelle est la fonction de la bibliothèque standard du langage C qui permet d'allouer dynamiquement de la mémoire et l'initialiser directement avec des 0 ?

- ☐ `free`
- ☐ `malloc`
- ☐ `realloc`
- ☐ `calloc`

Question 3 Faire un schéma mémoire et indiquez ce qu'affiche l'exécution de ce programme (Répondez sur la copie d'examen) :

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

int main() {
    char str[] = "O VALLEY OF THE APPRENTICES";
    char *copy = malloc((strlen(str) + 1) * sizeof(char));
    strcpy(copy, str);
    char str2[] = "TEACHER";
    char *ptr, **ptr2;

    str[1] = str[0];
    str[0] = 'T';
    str[2] = str[strlen(str)-1];
    ptr = str + 3;
    *ptr = *(ptr-1);
    ptr++;
    *ptr = *(ptr + 4);
    ptr = ptr + 1;
    *ptr = *ptr - 11;
    *(ptr+1) = *(ptr-1);
    *(ptr+2) = 'C';
    *(ptr+3) = *(ptr+4);
    *(ptr+4) = str[strlen(str)-4];
    ptr += 5;
    *ptr = *ptr + 8;
    ++ptr;
    ptr2 = &ptr;
    **ptr2 = *(str + 4);
    (*ptr2)++; (*ptr2)++;
    **ptr2 = (**ptr2) + 7;
    *(+++ptr2) = ' ';
    ptr++;
    *ptr = 'Y';
    ptr++;
    *((*ptr2)++) = str[1];
    *((*ptr2)++) = str[1] + 6;
    **ptr2 = *(ptr+1);
    ptr++;
    **ptr2 = ' ';
    ptr++;
    strcpy(ptr, str2);

    printf("Nouvelle chaîne de caractères :\n%s\n%s\n", str, copy);
}
```

Extrait du manuel pour les fonctions `strcpy` et `strlen` :

```
char *strcpy(char *dest, const char *src);

La  fonction strcpy() copie la chaîne pointée
par src, y compris le caractère nul (« \0 »)
final dans la chaîne pointée par dest.
La chaîne dest doit être assez grande pour
accueillir la copie.
```

```
size_t strlen(const char *s);
(size_t est un type représentant un entier
non signé, donc strictement positif)

La  fonction strlen() calcule la longueur
de la chaîne de caractères s, sans
compter l'octet nul (« \0 ») final.
```

..... ☐0 ☐1 ☐2 ☐3 ☐4

Question 4 Le code suivant est compilé avec la commande `gcc main2.c` et on l'exécute, si possible, avec `./a.out`. Que se passe-t-il ?

```
#include <stdio.h>

int f(int i) {
    i = i * 10;
    return i;
}

int main() {
    int i = 5;
    f(i);
    printf("%d\n", i);
}
```

- ☐ Erreur de segmentation
- ☐ Affiche `5`
- ☐ Affiche `50`
- ☐ Ça ne compile pas

Question 5 On souhaite allouer de l'espace mémoire pour 150 entiers. Quelle est la bonne instruction ?

- ☐ `int *buffer = malloc(150 * int);`
- ☐ `int buffer = malloc(150 * sizeof(int));`
- ☐ `int *buffer = 150 * malloc(int);`
- ☐ `int *buffer = malloc(150 * sizeof(int));`

Question 6 Soit la déclaration `Point2D p;` et la définition de structure suivante, quel est l'instruction permettant d'accéder au champ `x` ?

```
typedef struct {
    int id;
    float x;
    float y;
} Point2D;
```

- ☐ `p(x)`
- ☐ `p->x`
- ☐ `p.x`
- ☐ `x(p)`

Question 7 Soit les déclarations `int *ptr_i;` et `int i;`, on souhaite faire pointer `ptr_i` sur `i`. Quelle instruction permet cela ?

- ☐ `ptr_i = i*`
- ☐ `i = &ptr_i`
- ☐ `ptr_i = *i`
- ☐ `ptr_i = &i`

Question 8 ♣ Cochez les affirmations correctes :

- ☐ Le préprocesseur C est un programme qui assure une phase préliminaire à la compilation. Il permet notamment de définir des macros à l'aide de directives commençant par le caractère `"#"`.
- ☐ La désallocation d'un espace mémoire de la pile est automatique.
- ☐ Essayer de libérer une zone mémoire (avec la fonction `free`) déjà libérée auparavant n'entraine pas d'erreur.
- ☐ Un pointeur contient l'adresse d'une donnée dans la mémoire.
- ☐ Le symbole pour déclarer un pointeur est le même que pour déréférencer un pointeur.
- ☐ Les données stockées dans une zone mémoire allouées dynamiquement se retrouvent dans la pile.
- ☐ Il n'est pas possible de stocker l'adresse d'un pointeur de pointeur sur entier.
- ☐ Le langage C permet de façon simple d'appliquer les principes de la programmation orientée objet (classe, encapsulation, héritage/composition, polymorphisme).
- ☐ *Aucune de ces réponses n'est correcte.*

Question 9 En Shell, sachant que la variable `i` a été déclarée et initialisée en faisait `i=17`, que fait la commande `[ 15 -lt $i ]` ?

- ☐ Il y a une erreur de syntaxe
- ☐ Renvoie un code de retour -1
- ☐ Renvoie un code de retour 0
- ☐ Renvoie un code de retour 1

Question 10 Quelle commande permet de trouver dans l'arborescence du dossier `/home/user/Images` tous les fichiers dont le nom commence par "IMG" (uniquement ces fichiers, sans les répertoires) ?

- ☐ find /home/user/Images -name 'IMG*' -type d
- ☐ find /home/user/Images -name 'IMG*' -type f
- ☐ `ls /home/user/Images | grep ^IMG`
- ☐ `ls /home/user/Images | grep IMG`

Question 11 (Inspiré d'un exercice proposé par Christophe Bouthier)
Remplir la grille de « mots croisés » version Regex suivante. Le but est de remplir chaque case par un caractère qui correspond à la fois à l'expression régulière verticale et à l'expression régulière horizontale.

Exemple : Le caractère qui correspond à la fois à la regex vertical (`[AB]`) et à la regex horizontale (`[AZ]`) est `'A'`

	[AB]
[AZ]	A

..... ☐0 ☐1 ☐2 ☐3 ☐4

	[^35029]	**2|6|8|4|3**
0|9|2|5|1		
[03579]		

Question 12 Même question que la précédente.

..... ☐0 ☐1 ☐2 ☐3 ☐4

	[filorux03AZ5]	**3|5|a|d|g|j**
7|B|A|a|1		
[^AZ0-9ae-qs-z]		

Question 13 ♣ Cochez les affirmations correctes :

- ☐ Pour rediriger la sortie d'une commande vers un fichier en écrasant son contenu, on utilise `>>`
- ☐ L'option `-d` (`--delimiter`) de la commande `cut` peut s'utiliser sans l'option `-f` (`--fields`)
- ☐ `cat file.log | grep 'user : mallory'` est équivalent à `grep 'user : mallory' file.log`
- ☐ La commande `find` effectue une recherche récursive à partir du répertoire donné en argument.
- ☐ Dans un script Bash, la structure conditionnelle `if` s'utilise en remplacement de la commande `test` ou `[`
- ☐ En Bash, seul la structure de boucles `for` existe.
- ☐ En bash, la structure de boucle `for` est de type "pour chaque" (*for each* en anglais)
- ☐ *Aucune de ces réponses n'est correcte.*