



QCM

Langage C

QCM Langage C - Année 2020-2021
ENSEEIH, 1SN
Katia Jaffrès-Runser.

Examen Session 1
21 janvier 2021

Durée : 60 minutes.

Les réponses sont attendues SUR LA DERNIERE PAGE DU SUJET qui est à rendre. Les réponses données sur les autres feuilles du sujet ne sont pas prises en compte lors de l'évaluation.

Les questions faisant apparaître le symbole ♣ peuvent présenter zéro, une ou plusieurs bonnes réponses. Les autres ont une unique bonne réponse.

Une question simple rapporte au maximum 1 point, une question multiple au maximum 3 points. Des points négatifs pourront être affectés à de *très mauvaises* réponses.

Pour valider un choix, il faut complètement **NOIRCIR** la case.
Seules les cases sont analysées, il vous est donc possible d'écrire ailleurs sans incidence sur votre rendu.



1 Structure d'un programme, constantes et types.

On considère le programme suivant qui calcule le périmètre d'un cercle connaissant son rayon :

```
1  int main(){
2      #include <stdlib.h>
3      #include <stdio.h>
4      float rayon = 3.4; //rayon du cercle
5      char unite = 'c';
6      const float PI 3.1415;
7      printf("Le périmètre du cercle de rayon %f%c est %f%c\n",
8          rayon, unite, 2*PI*rayon, unite);
9      return EXIT_SUCCESS;
10 }
```

Question 1 ♣ Quelle(s) instruction(s) ne sont pas correctes :

- | | |
|--|---|
| <input checked="" type="checkbox"/> L'instruction de la ligne 2. | <input type="checkbox"/> L'instruction de la ligne 5. |
| <input type="checkbox"/> L'instruction de la ligne 9. | <input type="checkbox"/> Aucune de ces réponses n'est correcte. |
| <input checked="" type="checkbox"/> L'instruction de la ligne 6. | |

Question 2 Comment définir la constante préprocesseur MAX qui vaut 10 ?

- | | |
|--|---|
| <input type="checkbox"/> #const MAX 10 | <input checked="" type="checkbox"/> #define MAX 10; |
| <input type="checkbox"/> #define MAX = 10; | <input type="checkbox"/> #DEFINE MAX = 10 |
| <input type="checkbox"/> #const MAX = 10; | <input checked="" type="checkbox"/> #define MAX 10 |

Question 3 Comment déclarer un nouveau type t_tab qui est un tableau de 30 booléens ?

- | | |
|---|---|
| <input type="checkbox"/> define bool[30] t_tab; | <input checked="" type="checkbox"/> typedef bool t_tab[30]; |
| <input type="checkbox"/> typedef bool t_tab 30; | <input type="checkbox"/> define bool t_tab[30]; |

Question 4 Comment définir Reel comme un alias sur le type long float ?

- | | |
|--|---|
| <input checked="" type="checkbox"/> typedef long float Reel; | <input type="checkbox"/> #define long float Reel |
| <input type="checkbox"/> typedef Reel long float; | <input type="checkbox"/> type Reel is new long float; |

2 Variables et expressions

Question 5 Quelle est la valeur de a après l'exécution des instructions suivantes :

```
int a = 10;
int b = 3;
a -= b;
```

- | | |
|--|---------------------------------------|
| <input type="checkbox"/> Le programme ne compile pas | <input checked="" type="checkbox"/> 7 |
| <input type="checkbox"/> -3 | <input type="checkbox"/> 10 |
| <input type="checkbox"/> 3 | <input type="checkbox"/> 13 |

Question 6 ♣ Comment déclarer et initialiser une variable x de type double à 20 ?

- | | |
|--|---|
| <input type="checkbox"/> double x += 20.0; | <input checked="" type="checkbox"/> double x = 20; |
| <input checked="" type="checkbox"/> double x = 20.0; | <input type="checkbox"/> Aucune de ces réponses n'est correcte. |



Question 7 Quelles valeurs donner à XX1, XX2, XX3 et XX4 pour que le message « Bravo ! » s'affiche lors de l'exécution des instructions suivantes :

```
assert(XX1 == 3 * 5 - 2 * 5);
assert(XX2 == 32 % 10);
assert(XX3 == 32 / 10);
assert(XX4 == 32 / 10.0);
printf("%s", "Bravo !");
```

☐ A XX1=5, XX2=3, XX3=2 et XX4=3.2

☐ C XX1=10, XX2=3, XX3=2 et XX4=3

☒ B XX1=5, XX2=2, XX3=3 et XX4=3.2

☐ D XX1=5, XX2=2, XX3=3.2 et XX4=3.2

Question 8 Comment déclarer une variable n de type entier ?

☐ A n: int

☒ B int n;

☐ C int n

☐ D n of int;

Question 9 ♣ Cet exercice s'intéresse au concept de masquage des variables. Soit le programme suivant (les inclusions de bibliothèque sont volontairement omises) :

```
1  int main() {
2      int alea = 20;
3      int diviseur = 2;
4      float res1;
5      {
6          int alea = 3;
7          float diviseur = 2.0;
8          res1 = alea / diviseur;
9      }
10     int res2 = alea / diviseur;
11     return EXIT_SUCCESS;
12 }
```

Quelle(s) assertion(s) sont vraies si elles sont exécutées entre l'instruction 10 et 11 ?

☐ A assert(res2 == 1);

☒ B assert(res2 == 10);

☐ C assert(res1 == 10.0);

☒ D assert(res1 == 1.0);

☒ E assert(res1 == 1.5);

☐ F Aucune de ces réponses n'est correcte.

3 Pointeurs

Question 10 On suppose les instructions suivantes :

```
int var1 = 10;
int *p1 = &var1;
int *p2 = p1;
```

Quelle est la valeur de la variable p2 ?

☒ A l'adresse de var1.

☐ B 10

☐ C l'adresse de p1.



Question 11 ♣ Comment déclarer deux variables `a` et `b` de type pointeur sur réel ?

- | | |
|--|---|
| <input type="checkbox"/> <code>float* a, b;</code> | <input checked="" type="checkbox"/> <code>float *a, *b;</code> |
| <input checked="" type="checkbox"/> <code>float * a, * b;</code> | <input type="checkbox"/> Aucune de ces réponses n'est correcte. |
| <input checked="" type="checkbox"/> <code>float* a; float *b;</code> | |

4 Entrées/sorties

Question 12 ♣ La fonction `scanf` permet de lire des données typées entrées au clavier. Dans la suite, cocher les utilisations correctes de cette fonction. On supposera que les variables suivantes sont définies comme suit :

```
float prix; int val; char unite = 'm'; float peri; char nom[10];
```

- | | |
|--|--|
| <input type="checkbox"/> <code>scanf("%1.2d", peri);</code> | <input type="checkbox"/> <code>scanf("%s", &unite);</code> |
| <input checked="" type="checkbox"/> <code>scanf("%s", nom);</code> | <input checked="" type="checkbox"/> <code>scanf("%d %c", &val, &unite);</code> |
| <input type="checkbox"/> <code>scanf("%s", &nom);</code> | <input checked="" type="checkbox"/> <code>scanf("%f", &prix);</code> |
| <input checked="" type="checkbox"/> <code>scanf("Le prix est %f", &prix);</code> | <input type="checkbox"/> Aucune de ces réponses n'est correcte. |

Question 13 La fonction `printf` permet d'écrire des données typées à l'écran. Dans la suite, cocher l'affichage que l'on observe à l'écran si les instructions suivantes sont exécutées (on suppose que la bibliothèque `stdio.h` est connue) :

```
float prix = 1.4; int quantite = 100; char devise = 'E';  
printf("L'achat de %d %s revient à %1.3f%c.",  
      quantite, "ananas", quantite * prix, devise);
```

- | | |
|--|---|
| <input type="checkbox"/> L'achat de 100 ananas revient à 100 * 1.4E. | <input checked="" type="checkbox"/> L'achat de 100 ananas revient à 140.000E. |
| <input type="checkbox"/> L'achat de 100 ananas revient à 140E. | <input type="checkbox"/> L'achat de 100 "ananas" revient à 140.000E. |

5 Structures de contrôle

Question 14 On suppose que `sequence1`, `sequence2` et `sequence3` représentent plusieurs instructions où chaque instruction se termine par un point-virgule. On suppose aussi que `cond1` et `cond2` sont des expressions booléennes. La conditionnelle :

```
Si cond1 Alors sequence1 SinonSi cond2 Alors sequence2 Sinon sequence3
```

peut alors s'écrire en C :

- | |
|--|
| <input type="checkbox"/> <code>if (cond1) then { sequence1 } else if (cond2) then { sequence2 } else { sequence3 }</code> |
| <input checked="" type="checkbox"/> <code>if (cond1) { sequence1 } else if (cond2) { sequence2 } else { sequence3 }</code> |
| <input type="checkbox"/> <code>if (cond1) { sequence1 } elif (cond2) { sequence2 } else { sequence3 }</code> |

**Question 15 ♣**

Voici une fonction qui illustre la conditionnelle Selon :

```
int f(int n) {
    int r = 0;
    switch (n) {
        case 1:
            r += 1;
            break;
        case 2:
        case 3:
            r += 8;
            break;
        case 4:
            r += 10;
            break;
        case 7:
            r += 10;
            break;
        case 10:
            r += 100;
            break;
        case 13:
            r += 100;
            break;
        default:
            r -= 1;
    }
    return r;
}
```

Elle retourne une valeur de r qui dépend du paramètre n . Quels tests sont corrects parmi les propositions suivantes :

- ☐ A `assert(100 == f(13));` ☒ B `assert(99 == f(10));`
☒ C `assert(8 == f(3));` ☒ D `assert(-1 == f(12));`
☐ E `assert(0 == f(0));` ☐ F *Aucune de ces réponses n'est correcte.*

Question 16 ♣ Quelle(s) formulation(s) de la boucle `pour` sont juste(s) ? On supposera que `sequence` représente plusieurs instructions.

- ☐ A `for (int j = 1; j +=2 ; j < 10){ sequence }`
☒ B `for (int i = 0, i < 10, i += 2){ sequence }`
☒ C `for (int k = 100; k >= 0; k--){ sequence }`
☒ D `for (int j = 0; j <= 10 ; j += 2){ sequence }`
☐ E *Aucune de ces réponses n'est correcte.*

6 Énumération, Enregistrement et Tableau

Question 17 On souhaite définir un type énuméré qui représente les couleurs d'un jeu de cartes UNO rouge, vert, bleu, jaune. Quelle proposition retiendriez-vous pour cela ?

- ☒ A `enum Couleur {RGE, VER, BLE, JAU};`
☐ B `Couleur is enum {RGE, VER, BLE, JAU};`
☐ C `Couleur is new enum (RGE, VER, BLE, JAU);`

Question 18 Quelle proposition est vraie pour un type `enum Etat` qui représente les quatre états de la matières LIQ, SOL, GAZ, PLA dans cet ordre :

- ☒ A L'instruction `enum Etat etat1 = 1;` compile.
☐ B On a la relation d'ordre suivante : `LIQ > SOL > GAZ > PLA`.
☐ C LIQ, SOL, GAZ, PLA sont des chaînes de caractère.



Question 19 On souhaite définir un type enregistrement qui représente une carte à jouer caractérisée par une enseigne de type `enum Enseigne` et une valeur entière. Quelle proposition retiendriez-vous pour cela ?

- ☐ A `Carte is struct (enum Enseigne ens; int val;);`
☒ B `struct Carte { enum Enseigne ens; int val; };`
☐ C `struct Carte is new {enum Enseigne ens; int val;};`

Question 20 ♣ Le pointeur `ptr_carte` est initialisé de la façon suivante :

```
struct Carte carte1 = {PIQ, 9};  
struct Carte * ptr_carte = &carte1;
```

On souhaite afficher la valeur de `carte1` via le pointeur `ptr_carte`. Quelle(s) instruction(s) permet(tent) de le faire ?

- ☐ A `printf("%d", ptr_carte.All.val);` ☒ B `printf("%d", ptr_carte -> val);`
☐ C `printf("%d", ptr_carte.val);` ☐ D *Aucune de ces réponses n'est correcte.*

Question 21 On souhaite définir **un type** tableau qui représente un jeu de 52 cartes à jouer. Quelle proposition retiendriez-vous pour cela ?

- ☐ A `struct Carte Jeu[52];`
☒ B `typedef struct Carte Jeu[52];`
☐ C `typedef struct Carte[52] Jeu;`

Question 22 ♣ On souhaite initialiser la première carte d'un jeu avec l'as de pique. On suppose que la variable `jeu1` est déclarée comme suit `Jeu jeu1;`. Quelle(s) proposition(s) sont possibles ?

- ☒ A `jeu1(0).ens = PIQ; jeu1(0).val = 1;` ☒ B `jeu1->ens = PIQ; jeu1->val = 1;`
☐ C `jeu1[1].ens = PIQ; jeu1[1].val = 1;` ☐ D *Aucune de ces réponses n'est correcte.*

7 Sous-programmes

Question 23 On considère la portion de code suivante :

```
void p1(char *a) {  
    ...  
}  
void p2(char *b) {  
    char c;  
    XXX  
}
```

Cocher la valeur de `XXX` qui correspond à un appel possible de `p1`.

- ☐ A `p1(c);` ☒ B `p1(&b);`
☒ C `p1(b);`

Question 24 Quelle est la signature qui correspond à une procédure `p` qui prend comme premier paramètre un entier `n` en mode In et comme deuxième paramètre un entier `d` en In/Out.

- ☐ A `int p(int n, int d);` ☒ B `void p(int n, int *d);`
☐ C `void p(int* n, int *d);`



Question 25 Soit le programme suivant :

```
#include <stdio.h>
int f1(int* valeur) {
    *valeur = *valeur + 20;
    return *valeur;
}
int main(){
    int donnee = 20;
    int donnee_retournee = f1(&donnee);
    printf("donnee : %i ", donnee);
    printf("donnee_retournee : %i ", donnee_retournee);
}
```

Quelles sont les valeurs de `donnee` et `donnee_retournee` à la fin du programme principal ?

- ☐ `donnee = 40` et `donnee_retournee = 20` ☒ `donnee = 40` et `donnee_retournee = 40`
☐ `donnee = 20` et `donnee_retournee = 20`

8 Allocation dynamique

Question 26 On veut pouvoir enregistrer 10 entiers de plus dans un tableau de T entiers, tableau alloué dynamiquement. Un étudiant propose cette instruction qui compile et s'exécute sans erreur :

```
int *tab = realloc(tab, (T+10)*sizeof(int));
```

Pourquoi cet étudiant se trompe-t-il ?

- ☐ Il faut indiquer uniquement `10 * sizeof(int)` en second paramètre de l'appel à `realloc`.
☒ En cas d'échec de la réallocation, `realloc` retourne `NULL` et on aura perdu l'adresse de la mémoire initiale dans `tab`.

Question 27 ♣ Cocher la ou les instructions correctes qui permettent d'allouer de l'espace pour enregistrer un réel avec `malloc`.

- ☐ `float v = malloc(sizeof(float));` ☒ `float *v = malloc(sizeof(*v));`
☒ `float *v = malloc(sizeof(float));` ☐ *Aucune de ces réponses n'est correcte.*
☐ `float *v = malloc(float);`

Question 28 Comment savoir si l'allocation dynamique a réussi ?

- ☒ On vérifie si l'allocateur ne retourne pas le pointeur `NULL`.
☐ L'exécution se déroule sans erreur.

Question 29 ♣ On souhaite allouer *** dynamiquement *** une variable tableau de 15 caractères. Cocher la ou les bonne(s) instruction(s) :

- ☒ `char *t = malloc(15 * sizeof(char));`
☒ `char *t = calloc(15, sizeof(char));`
☒ `char *t = malloc(15 * sizeof(*t));`
☐ `char *t = 15 * malloc(sizeof(char));`
☐ *Aucune de ces réponses n'est correcte.*



Question 30 ♣ Soient les instructions suivantes :

```
int *valeur = malloc(sizeof(int));
*valeur = 10;
free(valeur);
printf("la valeur est %d", *valeur);
```

Qu'affiche la dernière instruction `printf` ?

- ☐ A 10 ☒ Probablement 10
- ☒ B Ce code est faux. ☐ E Aucune de ces réponses n'est correcte.
- ☒ C L'exécution échoue à cause d'une erreur de segmentation

Question 31 Voici la définition de la procédure `malloc` :

```
void* malloc(size_t taille);
```

A quoi sert cette procédure ?

- ☐ A A allouer une zone mémoire de `taille` bits
- ☒ B A allouer une zone mémoire de `taille` octets
- ☐ C A allouer une zone mémoire de `size_t` octets

Question 32 ♣ L'allocateur `realloc` permet de modifier la taille mémoire allouée dynamiquement à une adresse donnée. Voici sa signature :

```
void* realloc(void* ptr_mem, size_t taille)
```

Cocher la ou les propositions justes :

- ☒ A `ptr_mem` contient l'adresse de la zone mémoire après réallocation (mode in out).
- ☐ B `taille` représente l'incrément de taille mémoire demandé.
- ☒ C Si `ptr_mem` vaut `NULL`, `realloc` se comporte comme `malloc`.
- ☒ D `realloc` retourne `NULL` ou l'adresse d'une zone mémoire de `taille` octets.
- ☐ E Aucune de ces réponses n'est correcte.

Question 33 Soient les instructions suivantes :

```
char *initiale = malloc(sizeof(char));
*initiale = 'A';
```

Cocher l'instruction permettant de libérer la mémoire :

- ☐ A `initiale = NULL;` ☐ C `initiale.free();`
- ☒ B `free(initiale); initiale = NULL;` ☐ D `free(initiale, sizeof(char));`



Question 34 ♣ Pour le jeu d'instructions suivant :

```
enum outil {BECHE, PELLE, SEAU, ARROSOIR};
enum outil *ustensile;
ustensile = calloc(1, sizeof(enum outil));
assert(*ustensile == XXX);
```

Cocher une valeur pour XXX qui valide l'assert.

☐ A NULL
☒ B 0

☒ C BECHE
☐ D Aucune de ces réponses n'est correcte.

9 Les modules

Question 35 On souhaite définir un module `pile` en C. Quels fichiers doit-on créer par convention ?

☐ A Pour l'interface `pile.c` et `pile.h` pour le corps
☐ B Pour l'interface `pile.h` et `pile.cc` pour le corps
☒ C Pour l'interface `pile.h` et `pile.c` pour le corps

Question 36 Est-ce que la commande suivante produit un exécutable ? On suppose qu'il n'y a pas d'erreur dans les programmes.

```
c99 -Wextra -pedantic afficher.c date.c -o afficher
```

☒ A Oui, si un sous-programme `int main()` existe dans un des deux fichiers. ☐ B Non

Question 37 Le corps du module `date.c` présente la fonction suivante :

```
static int max(int a, int b) {
    if (a > b) {
        return a;
    } else {
        return b;
    }
}
```

Que doit-on faire pour que le programme principal `visualiser.c` qui inclut `date.h` puisse utiliser `max(x, y)` ?

☒ A Il faut spécifier `max` dans `date.h` et supprimer le mot-clé `static`.
☐ B Il faut spécifier `max` dans `date.h` et rajouter une garde conditionnelle dans `date.h`



Question 38 L'interface du module `date.h` présente la structure suivante :

```
#ifndef DATE__H
#define DATE__H
struct Date {
    int jour;
    int mois;
};
# endif
```

Quel est le rôle des commandes pré-processeur `#ifndef`, `#define` et `#endif` ?

- ☐ A De définir un type `struct Date` que le pré-processeur peut exploiter.
- ☒ B De pouvoir compiler même si `date.h` est inclus plusieurs fois.

Question 39 Un programmeur souhaite utiliser son propre module `pile` dans son programme principal décrit dans le fichier `principal.c`. Quelle instruction doit-on ajouter au début de `principal.c` ?

- ☐ A `#include <pile.h>`
- ☐ B `#import "pile.h"`
- ☒ C `#include "pile.h"`

10 Make

Question 40 ♣ Soit la règle suivante :

```
a:b c
xxx
```

La commande `xxx` sera exécutée :

- ☒ A si `a` n'existe pas [et `b` et `c` existent]
- ☒ B si `b` est plus récent que `a`
- ☐ C si `a` n'existe pas [et `b` et `c` n'existent pas]
- ☐ D Aucune de ces réponses n'est correcte.

Question 41 Voici une des règles explicites listées dans un `makefile` :

```
date.o: date.c date.h
c99 -Wextra -pedantic -c date.c
```

Quelle est la commande exécutée par cette règle ?

- ☐ A `date.o: date.c date.h`
- ☒ B `c99 -Wextra -pedantic -c date.c`



Question 42 Les premières règles d'un fichier `Makefile` sont les suivantes :

```
all: test_file exemple_file

test_file: test_file.o file.o
    c99 test_file.o file.o -o test_file

exemple_file: exemple_file.o file.o
    c99 exemple_file.o file.o -o exemple_file
```

Quel(s) fichier(s) génère la commande `make all` ? On supposera que `make` n'a jamais été lancé.

- ☐ A test_file, exemple_file
- ☒ B test_file, exemple_file, test_file.o, file.o, exemple_file.o
- ☐ C test_file, test_file.o, file.o



+50/12/55+



+50/13/54+

Feuille de réponses :

Nom et prénom :

BLANCHON Julien

Consignes :

Les réponses aux questions sont à donner exclusivement sur cette feuille : les réponses données sur les feuilles précédentes ne seront pas prises en compte.

Seules les cases sont analysées, il vous est donc possible d'écrire ailleurs sans incidence sur votre rendu.

- Question 1 : ☒ A ☒ B ☒ C ☐ D ☐ E
- Question 2 : ☐ A ☐ B ☐ C ☒ D ☐ E ☒ F
- Question 3 : ☐ A ☐ B ☒ C ☐ D
- Question 4 : ☒ A ☐ B ☐ C ☐ D
- Question 5 : ☐ A ☐ B ☐ C ☒ D ☐ E ☐ F
- Question 6 : ☐ A ☒ B ☒ C ☐ D
- Question 7 : ☐ A ☒ B ☐ C ☐ D
- Question 8 : ☐ A ☐ B ☒ C ☐ D
- Question 9 : ☐ A ☐ B ☒ C ☒ D ☒ E ☐ F
- Question 10 : ☒ A ☐ B ☐ C
- Question 11 : ☐ A ☒ B ☒ C ☒ D ☐ E
- Question 12 : ☐ A ☒ B ☐ C ☒ D ☐ E ☒ F ☒ G ☒ H
- Question 13 : ☐ A ☐ B ☒ C ☐ D
- Question 14 : ☐ A ☒ B ☐ C
- Question 15 : ☐ A ☒ B ☐ C ☒ D ☒ E ☐ F
- Question 16 : ☐ A ☒ B ☒ C ☒ D ☐ E
- Question 17 : ☒ A ☐ B ☐ C
- Question 18 : ☒ A ☐ B ☐ C
- Question 19 : ☐ A ☒ B ☐ C
- Question 20 : ☐ A ☐ B ☒ C ☐ D
- ~~Question 21 : ☐ A ☒ B ☐ C~~ 13
- Question 22 : ☒ A ☐ B ☒ C ☐ D
- Question 23 : ☐ A ☒ B ☒ C
- Question 24 : ☐ A ☐ B ☒ C
- Question 25 : ☐ A ☐ B ☒ C
- Question 26 : ☐ A ☒ B
- Question 27 : ☐ A ☒ B ☐ C ☒ D ☐ E
- Question 28 : ☒ A ☐ B
- Question 29 : ☒ A ☒ B ☒ C ☐ D ☐ E
- Question 30 : ☐ A ☒ B ☒ C ☒ D ☐ E



+50/14/53+

- Question 31 : ☐ A ☒ B ☐ C
- Question 32 : ☒ A ☐ B ☒ C ☒ D ☐ E
- Question 33 : ☐ A ☒ B ☐ C ☐ D
- Question 34 : ☐ A ☒ B ☒ C ☐ D
- Question 35 : ☐ A ☐ B ☒ C
- Question 36 : ☒ A ☐ B
- Question 37 : ☒ A ☐ B
- Question 38 : ☐ A ☒ B
- Question 39 : ☒ A ☒ B ☒ C ☐ D → ? C
- Question 40 : ☒ A ☒ B ☐ C ☐ D
- Question 41 : ☐ A ☒ B
- Question 42 : ☐ A ☒ B ☐ C



+50/16/51+