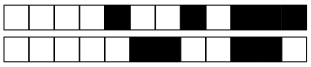

QCM Langage C - Année 2020-2021
ENSEEIHT, 1SN
Katia Jaffrès-Runser.

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 :

- [ ] A L'instruction de la ligne 5.
- [ ] B L'instruction de la ligne 9.
- [x] L'instruction de la ligne 6.
- [x] L'instruction de la ligne 2.
- [ ] E *Aucune de ces réponses n'est correcte.*

Question 2 Comment définir la constante préprocesseur DEBUG qui vaut 0 ?

- [ ] A `#DEFINE DEBUG = 0`
- [x] `#define DEBUG 0`
- [ ] C `#define DEBUG 0;`
- [ ] D `#const DEBUG = 0;`
- [ ] E `#define DEBUG = 0;`
- [ ] F `#const DEBUG 0`

Question 3 Comment déclarer un nouveau type `t_tab` qui est un tableau de 20 réels ?

- [x] `typedef double t_tab[20];`
- [ ] B `typedef double t_tab 20;`
- [ ] C `define double[20] t_tab;`
- [ ] D `define double t_tab[20];`

Question 4 Comment définir `Entier` comme un alias sur le type `int` ?

- [ ] A `#define int Entier`
- [ ] B `typedef Entier int;`
- [x] `typedef int Entier;`
- [ ] D `type Entier is new int;`

2 Variables et expressions

Question 5 ♣ Comment déclarer et initialiser une variable `test` de type `bool` à `true` ?

- [x] `bool test = true;`
- [x] `bool test = 1;`
- [ ] C `bool test <- true;`
- [ ] D *Aucune de ces réponses n'est correcte.*

Question 6 Quelle est la valeur de `a` après l'exécution des instructions suivantes :

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

- [ ] A Le programme ne compile pas
- [ ] B 10
- [ ] C 100
- [ ] D 13
- [x] 30
- [ ] F 3

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 == 5 - 2 * 5);
assert(XX2 == 25 % 10);
assert(XX3 == 25 / 10);
assert(XX4 == 25 / 10.0);
printf("%s", "Bravo !");
```

- [X] `XX1=-5, XX2=5, XX3=2 et XX4=2.5`
- [ ] B `XX1=-5, XX2=2, XX3=5 et XX4=2.5`
- [ ] C `XX1=15, XX2=5, XX3=2 et XX4=2`
- [ ] D `XX1=-5, XX2=5, XX3=2.5 et XX4=2.5`

Question 8 Comment déclarer une variable `x` de type réel ?

- [ ] A `float x`
- [X] `float x;`
- [ ] C `x of float;`
- [ ] D `x: float;`

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      {
5          int alea = 3;
6          float diviseur = 2.0;
7          float res = alea / diviseur;
8          assert(res == 1.5);
9      }
10     int res = alea / diviseur;
11     assert(res == 10);
12     printf("%s", "Les tests passent");
13     return EXIT_SUCCESS;
14   }
```

Quelle(s) variable(s) sont masquée(s) dans cet exemple ?

- [ ] A `float diviseur` à la ligne 6 est masquée.
- [X] `int diviseur` à la ligne 3 est masquée.
- [ ] C `int res` à la ligne 10 est masquée.
- [ ] D *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` ?

- [X] l'adresse de `var1`.
- [ ] B `10`
- [ ] C l'adresse de `p1`.

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

- [X] `char* a; char *b;`
- [X] `char *a, *b;`
- [ ] C `char* a, b;`
- [X] `char * a, * b;`
- [ ] E *Aucune de ces réponses n'est correcte.*

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];
```

- [ ] A `scanf("%s", &nom);`
- [X] `scanf("%s", nom);`
- [X] `scanf("%f", &prix);`
- [ ] D `scanf("%s", &unite);`
- [ ] E `scanf("Le prix est %f", &prix);`
- [ ] F `scanf("%1.2d", peri);`
- [X] `scanf("%d %c", &val, &unite);`
- [ ] H *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 = 0.6; int quantite = 20; char devise = 'p';
printf("L'achat de %d %s revient à %1.1f%c.",
                        quantite, "oranges", quantite * prix, devise);
```

- [ ] A L'achat de 20 "oranges" revient à 12p.
- [ ] B L'achat de 20 oranges revient à 20 * 0.6p.
- [X] L'achat de 20 oranges revient à 12.0p.
- [ ] L'achat de 20 "oranges" revient à 12.0p.

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 :

- [ ] A `if (cond1) { sequence1 } elif (cond2) { sequence2 } else { sequence3 }`
- [X] `if (cond1) { sequence1 } else if (cond2) { sequence2 } else { sequence3 }`
- [ ] C `if (cond1) then { sequence1 } else if (cond2) then { sequence2 } else { sequence3 }`

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:
    case 7:
       r += 10;
    case 10:
    case 13:
       r += 100;
    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(1 == f(0));`
- [B] `assert(99 == f(6));`
- [X] `assert(8 == f(3));`
- [X] `assert(-1 == f(11));`
- [X] `assert(99 == f(13));`
- [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 }`
- [X] `for (int j = 0; j <= 10 ; j += 2){ sequence }`
- [X] `for (int k = 100; k >= 0; k--){ sequence }`
- [E] *Aucune de ces réponses n'est correcte.*

6 Enumération, 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 ?

- [X] `enum Couleur {RGE, VER, BLE, JAU};`
- [B] `Couleur is new enum (RGE, VER, BLE, JAU);`
- [C] `Couleur is enum {RGE, VER, BLE, JAU};`

Question 18 Quelle proposition est vraie pour un type `enum Couleur` qui définit les quatre couleurs `RGE, VER, BLE, JAU` dans cet ordre :

- [X] On a la relation d'ordre suivante : `RGE < VER < BLE < JAU`.
- [B] L'instruction `enum Couleur col = JAU;` est équivalente à `enum Couleur col = 4;`.
- [C] L'instruction `enum Couleur col = 0;` ne compile pas.

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 ?

- [X] `struct Carte { enum Enseigne ens; int val; };`
- [ ] B `Carte is struct (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);`
- [X] `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[52] Jeu;`
- [X] `typedef struct Carte Jeu[52];`

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

7 Sous-programmes

Question 23 Quelle est la signature qui correspond à une procédure `p` qui prend comme premier paramètre un entier `n` en mode `Out` et comme deuxième paramètre un entier `d` en In/Out.

- [ ] A `void p(int n, int d);`
- [X] `void p(int *n, int *d);`
- [ ] C `int p(int n, int d);`

Question 24 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(&b);`
- [X] `p1(b);`
- [ ] C `p1(c);`

Question 25 Soit le programme suivant :

```
#include <stdio.h>
int f1(int* valeur) {
   *valeur = *valeur / 10;
   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 ?

- [X] `donnee = 2` et `donnee_retournee = 2`
- [ ] B `donnee = 20` et `donnee_retournee = 2`
- [ ] C `donnee = 20` et `donnee_retournee = 20`

8 Allocation dynamique

Question 26 ♣ Cocher la ou les instructions correctes qui permettent d'allouer de l'espace pour enregistrer un réel avec `malloc`.

- [ ] A `float v = malloc( sizeof(float));`
- [ ] B `float *v = malloc( float );`
- [X] `float *v = malloc( sizeof(*v));`
- [X] `float *v = malloc( sizeof(float));`
- [ ] E *Aucune de ces réponses n'est correcte.*

Question 27 ♣ On souhaite allouer ***** dynamiquement ***** une variable tableau de 10 entiers. Cocher la ou les bonne(s) instruction(s) :

- [ ] A `int *t = 10 * malloc(sizeof(int));`
- [X] `int *t = calloc(10, sizeof(int));`
- [X] `int *t = malloc(10 * sizeof(int));`
- [X] `int *t = malloc(10 * sizeof(*t));`
- [ ] E *Aucune de ces réponses n'est correcte.*

Question 28 Comment savoir si l'allocation dynamique a échoué dans un programme ?

- [X] On vérifie si l'allocateur retourne le pointeur `NULL`.
- [ ] B On observe un signal `segmetation fault` à l'exécution.

Question 29 ♣ 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 :

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

Question 30 ♣ Soient les instructions suivantes :

```
char *initiale = malloc(sizeof(char));
*initiale = 'A';
free(initiale);
printf("L'initiale est %c", *initiale);
```

Qu'affiche la dernière instruction `printf` ?

- [A] L'exécution échoue à cause d'une erreur de segmentation
- [X] Ce code est faux.
- [C] `'A'`
- [X] Probablement 'A'
- [E] *Aucune de ces réponses n'est correcte.*

Question 31 Soient les instructions suivantes :

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

Cocher l'instruction permettant de libérer la mémoire :

- [A] `free(initiale, sizeof(char));`
- [B] `initiale = NULL;`
- [X] `free(initiale); initiale = NULL;`
- [D] `initiale.free();`

Question 32 On veut pouvoir enregistrer 15 caractères de plus dans un tableau de `T` caractères, tableau alloué dynamiquement. Un étudiant propose cette instruction qui compile et s'exécute sans erreur :

```
char *str = realloc(str, (T+15)*sizeof(char));
```

Pourquoi cet étudiant se trompe-t-il ?

- [X] En cas d'échec de la réallocation, `realloc` retourne `NULL` et on aura perdu l'adresse de la mémoire initiale dans `tab`.
- [B] Il faut indiquer uniquement `15 * sizeof(char)` en second paramètre de l'appel à `realloc`.

Question 33 Voici la définition de la procédure `malloc` :

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

Quelle proposition caractérise le type `size_t` du paramètre `taille` ?

- [X] c'est un alias de `unsigned int`
- [ ] la taille mémoire demandée en octet
- [C] la taille mémoire demandée en bit

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.

- [X] `0`
- [B] `NULL`
- [X] `BECHE`
- [D] *Aucune de ces réponses n'est correcte.*

9 Les modules

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

- [ ] Pour l'interface `date.h` et `date.cc` pour le corps
- [ ] Pour l'interface `date.c` et `date.h` pour le corps
- [x] Pour l'interface `date.h` et `date.c` pour le corps

Question 36 Dans quelle partie du module date trouve-t-on typiquement le type d'instructions suivantes :

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

- [ ] Dans le corps `date.c`.
- [x] Dans l'interface `date.h`.

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;
  }
}
```

Le programme principal `visualiser.c` inclut `date.h`. Peut-il utiliser le sous-programme `max` dans `visualiser.c` ?

- [ ] Oui
- [x] Non
- [ ] Oui, mais seulement si la garde conditionnelle est présente dans `date.h`.

Question 38 Un programmeur souhaite utiliser son propre module `fraction` dans son programme principal décrit dans le fichier `principal.c`. Quelle instruction doit-on ajouter au début de `principal.c` ?

- [x] `#include "fraction.h"`
- [ ] `#include <fraction.h>`
- [ ] `#include "fraction"`

Question 39 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 -c liste.c
```

- [ ] Oui, si un sous-programme `int main()` existe dans `liste.c`.
- [x] Non

10 Make

Question 40 ♣ Soit la règle suivante :

```
a:b c
  xxx
```

La commande `xxx` sera exécutée :

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

Question 41 Une règle dans un `makefile` suit la structure suivante :

```
a:b
  c
```

Quels sont les termes qui décrivent respectivement les parties `a, b` et `c` d'une règle ?

- [ ] A dépendance, cible, commande
- [X] cible, dépendance, commande
- [ ] C commande, cible, dépendance

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, test_file.o, file.o`
- [X] `test_file, exemple_file`
- [X] `test_file, exemple_file, test_file.o, file.o, exemple_file.o`

+151/11/28+

Feuille de réponses :

Nom et prénom : OUKHNINI Hamid

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] [X] [X] [E]
Question 2 : [A] [X] [C] [D] [E] [F]
Question 3 : [X] [B] [C] [D]
Question 4 : [A] [B] [X] [D]
Question 5 : [X] [X] [C] [D]
Question 6 : [A] [B] [C] [D] [X] [F]
Question 7 : [X] [B] [C] [D]
Question 8 : [A] [X] [C] [D]
Question 9 : [A] [X] [C] [D]
Question 10 : [X] [B] [C]
Question 11 : [X] [X] [C] [X] [E]
Question 12 : [A] [X] [X] [D] [E] [F] [X] [H]
Question 13 : [A] [B] [X] [D]
Question 14 : [A] [X] [C]
Question 15 : [A] [B] [X] [X] [X] [F]
Question 16 : [A] [ ] [X] [X] [E] → corrector
Question 17 : [X] [B] [C]
Question 18 : [X] [B] [C]
Question 19 : [X] [B] [C]
Question 20 : [A] [X] [C] [D]
Question 21 : [A] [B] [X]
Question 22 : [A] [B] [X] [D]
Question 23 : [A] [X] [C]
Question 24 : [A] [X] [C]
Question 25 : [X] [B] [C]
Question 26 : [A] [B] [X] [X] [E]
Question 27 : [A] [X] [X] [X] [E]
Question 28 : [X] [B]
Question 29 : [X] [X] [C] [D] [E]
Question 30 : [A] [X] [C] [X] [E]

+151/12/27+

Question 31 : [A] [B] [X] [D]

Question 32 : [X] [B]

Question 33 : [X] [B] [C]

Question 34 : [X] [B] [X] [D]

Question 35 : [A] [B] [X]

Question 36 : [A] [X]

Question 37 : [A] [X] [C]

Question 38 : [X] [B] [C]

Question 39 : [A] [X]

Question 40 : [X] [X] [C] [D] → Corrector

Question 41 : [A] [X] [C]

Question 42 : [A] [B] [X]