Introduction au C
QCM d'auto évaluation
Année scolaire 2014-2015
Ensimag 1A

NOTA BENE :

- Le QCM a une vocation d'auto évaluation individuelle et ne fait l'objet d'aucune notation
- Essayez de comprendre les codes proposés « sur papier » pour commencer. Puis vérifiez votre réponse en exécutant ces codes sur machine
- Si vous rencontrez des problèmes, n'hésitez pas à communiquer au sein de votre groupe afin de vous expliquer les uns aux autres et de progresser ensemble
- Le QCM est à utiliser en liaison avec votre carte de compétences quotidienne
- On suppose pour toutes les questions qu'il y a assez de mémoire pour les allocations, et que les inclusions nécessaires (par exemple #include <stdio.h> ont été faites).

Question 1. Quelle valeur vaut x après l'exécution de ce code ?

```
int32_t x;
int32_t *p;
p = &x;
x = 40;
(*p)++;
```

40 41 42 Autre chose Erreur à la compilation ou à l'exécution

Question 2. Qu'affiche le code suivant en supposant que l'utilisateur entre 5 au clavier ?

```
int32_t x;
scanf("%d",&x);
if (x = 2) {
        printf("b");
}
else if (x = 4) {
        printf("d");
}
else {
        printf("z");
}
```

b d z Autre chose On ne peut pas savoir

Question 3. Que sait-on de la valeur affichée par le programme ci-dessous ?

```
uint32_t *n;
n = malloc(sizeof(uint32_t));

*n = 5;
free(n);
n = malloc(sizeof(uint32_t));
printf("%u\n", n);
```

Que c'est la valeur 0 Que c'est la valeur 5 Que ce n'est ni 0 ni 5 Rien

Question 4. En supposant que le tableau tab est à l'adresse 0xF000A000, qu'affiche le code ci-dessous sur une machine 64 bits ?

```
int32_t tab[] = {0, 1, 2, 3};
int32_t *p = tab + 4;
printf("0x%x\n", (uint32_t) p);
```

0xF000A000	0xF000A004	0xF000A008	0xF000A010
0xF000A016	0xF000A032	Autre chose	On ne peut pas savoir

Question 5. Qu'affiche le code suivant ?

```
typedef union _int_float {
        uint32_t mon_int;
        float mon_float;
} int_float;


int_float trois;
trois.mon_float = 3.0;
printf("%x\n", trois.mon_int);
```

3	3.0	0	Autre chose	On ne peut pas savoir

Question 6. Qu'affiche le code suivant :

```
int32_t a = 0xF0;
if (a & 0x10 != 0)
        printf("1\n");
else
        printf("2\n");
```

1	2	Erreur à la compilation ou à l'exécution

Question 7. Qu'affiche le code suivant ?

```
typedef struct _struct_type
{
        int32_t (*chat)(int32_t, int32_t);
        int32_t (*chien[2])();
} struct_type;

int32_t sup(int32_t a, int32_t b)
{
        return (a>b ? 0 : 1);
}

int32_t plus(int32_t c, int32_t d)
{
        return c + d;
}

int32_t moins(int32_t c, int32_t d)
{
        return c - d;
}

int32_t main()
{
        int32_t a = 20;
        int32_t b = 10;
        int32_t c = 30;
        int32_t d = 3;

        struct_type cible;

        cible.chat = sup;
        cible.chien[0]= &plus;
        cible.chien[1]= moins;

        printf("%d\n",cible.chien[(cible.chat)(a,b)](c,d));

        return 0;
}
```

Un résultat (le préciser) Erreur à la compilation ou à l'exécution

Question 8. Barrer dans le code suivant la (les) ligne(s) qui provoquent une erreur de compilation :

```
int32_t tab[10];
int32_t (*p)[10];
int32_t *p2;

p = &tab;
p2 = &tab;
```

Question 9. Barrer dans le code suivant les lignes qui provoquent une erreur de compilation :

```
uint32_t tab[10];
uint32_t *ptab = malloc(sizeof(uint32_t)*10);

ptab[1] = 1;
tab[0] = ptab[1];
ptab[1] = tab[2];
if (tab[0] == ptab[1]) {
        tab[0]++;
        ptab[0]++;
}
if (tab == ptab) {
        tab = ptab;
        ptab = tab;
}
tab++;
ptab++;
```

Question 10. Qu'affiche le code ci-dessous ?

```
char chaine[] = "toto";
printf("%p %p %p\n", chaine, &chaine, &chaine[0]);
```

Trois adresses différentes Trois adresses identiques
Deux sur trois identiques Erreur à la compilation ou à l'exécution
=> Expliquez pourquoi

Question 11. Qu'affiche le code ci-dessous ?

```
char *chaine = "toto";
printf("%p" %p %p\n", chaine, &chaine, &chaine[0]);
```

Trois adresses différentes Trois adresses identiques
Deux sur trois identiques Erreur à la compilation ou à l'exécution
=> Expliquez pourquoi

Question 12. Que sait-on du résultat affiché par le programme suivant :

```
void print_first(int32_t **tab)
{
        printf("%d\n", tab[0][0]);
}

int32_t main()
{
        int32_t tab[10][10];
        tab[0][0] = 5;
        print_first((int32_t **) tab);
}
```

Que c'est "0" Que c'est "5" Que ce n'est ni "0" ni "5" Rien
Erreur de compilation ou à l'exécution

Question 13. Que sait-on du résultat affiché par le programme suivant :

```
void print_first(int32_t tab[10][10])
{
      printf("%d\n",tab[0][0]);
}

int32_t main()
{
        int32_t **tab;
        int32_t i;
        tab = malloc(sizeof(int32_t*)*10);
        for (i = 0; i < 10; i++) {
                tab[i] = malloc(sizeof(int32_t)*10);
        }
        tab[0][0] = 5;
        print_first((int32_t (*)[10]) tab);
}
```

Que c'est "0" Que c'est "5" Que ce n'est ni "0" ni "5" Rien
Erreur de compilation ou à l'exécution

Question 14. Qu'affiche le code suivant ?

```
typedef enum _fruit {
        POIRE,
        POMME,
        PECHE
} fruit;

int32_t main()
{
        fruit mon_fruit;
        mon_fruit = POIRE;
        switch (mon_fruit) {
        case POIRE:
                printf("J'aime ça\n");
        case POMME:
                printf("J'aime un peu ça\n");
        case PECHE:
                printf("J'aime pas ça\n");
        }
        return 0;
}
```

"J'aime ça" "J'aime un peu ça" "J'aime pas ça" Rien Erreur à la compilation ou à l'exécution

Question 15. Qu'affiche le code suivant ?

```
typedef enum _fruit {
        POIRE,
        POMME,
        PECHE
} fruit;

int32_t main()
{
        fruit mon_fruit;
        mon_fruit = 42;
        switch (mon_fruit) {
        case POIRE:
                printf("J'aime ça\n");
        case POMME:
                printf("J'aime un peu ça\n");
        case PECHE:
                printf("J'aime pas ça\n");
        }
        return 0;
}
```

"J'aime ça" "J'aime un peu ça" "J'aime pas ça" Rien Erreur à la compilation ou à l'exécution

Question 16. Qu'affiche le code suivant ?

```
#define LA_REPONSE_ULTIME 42
#define CARRE(x) (x*x)

printf("%d\n", CARRE(LA_REPONSE_ULTIME-1));
```

-1 1 42 1681

Question 17. Qu'affiche le code suivant ?

```
#define LA_REPONSE_ULTIME 42

typedef struct _variable {
        char *name;
        int32_t value;
        struct _variable *suiv;
} variable_t;

typedef variable_t * variable;

int32_t main()
{
        variable *ma_variable = malloc(sizeof(variable));
        (*ma_variable)->value = LA_REPONSE_ULTIME;
        printf("La réponse ultime est : %d\n",(*ma_variable)->value);
}
```

0 1 42 Erreur à la compilation ou à l'exécution

Question 18. Qu'affiche le code suivant ?

```
void main()
{
        int x;
        while(true) {
                printf("Entrer un entier :");
                scanf("%d", &x);
                assert(x > 2);
                printf("Vous avez entré %d.\n", x);
        }
}
```

(1) - Entrer un entier : 2
Vous avez entré 2.
Entrer un entier: 4
Assertion failed: x, file main.c, line ...
Aborted.

(2) - Entrer un entier : 2
Assertion failed: x, file main.c, line ...
Aborted.

(3) - Entrer un entier : 2
Vous avez entré 2.
Entrer un entier : 4
Vous avez entré 4.

(1) (2) (3) Autre chose On ne peut pas savoir

Question 19. Quelle est la taille en mémoire de la structure secondaire, définie ci-après :

```
struct initial {
        int32_t a,b;
        float x;
};

struct secondaire {
        struct initial si;
        int32_t y;
        int32_t *k;
}
```

(1) - sizeof(secondaire) = sizeof(int32_t) + sizeof(float)

(2) - sizeof(secondaire) = sizeof(int32_t) + sizeof(int32_t *) + sizeof(struct initial *)

(3) - sizeof(secondaire) = 3*sizeof(int32_t) + sizeof(float) + sizeof(int32_t *)

(4) - sizeof(secondaire) = sizeof(int32_t) + sizeof(int32_t *) + sizeof(struct initial)

(1) (2) (3) (4)

Question 20. Un projet regroupe trois fichiers bib.h, bib.c et main.c

Le Makefile associé est:

```
mon_executable: bib.o main.o
           gcc -o mon_executable bib.o main.o

bib.o: bib.c
           gcc -o bib.o -c bib.c -Wall -O

main.o: main.c bib.h
           gcc -o main.o -c main.c -Wall -O
```

Quelles sont les commandes utilisées par make durant l'évaluation des règles ?

(1) - gcc -o bib.o -c bib.c -Wall -O
gcc -o main.o -c main.c -Wall -O
gcc -o mon_executable bib.o main.o

(2) - gcc -o mon_executable bib.o main.o
gcc -o bib.o -c bib.c -Wall -O
gcc -o main.o -c main.c -Wall -O

(3) - gcc -o main.o -c main.c -Wall -O
gcc -o bib.o -c bib.c -Wall -O
gcc -o mon_executable bib.o main.o

(1) (2) (3) Autre chose On ne peut pas savoir

Question 21. Soit le code suivant ?

```
void ecrire (char *fichier)
{
        FILE *fp;
        fp = fopen(fichier,"w");
        printf("Pointeur Fichier = %u\n", fp);
        fprintf(fp, "Bienvenue sur le fichier de sortie\n");
        fclose(fp);
}

int32_t main(int argc, char **argv)
{
        ecrire(argv[argc-2]);
        return 0;
}
```

On appelle le programme de la façon suivante : test sortie.txt
Quel résultat donne cette commande ?

(1) - Affichage de l'adresse de fp et écriture du texte "Bienvenue sur le fichier de sortie" dans le fichier "sortie.txt"
(2) - Ecriture de l'adresse de fp dans le fichier "sortie.txt" et affichage de "Bienvenue sur le fichier de sortie"

(1) (2) Erreur de compilation ou erreur à l'exécution

Question 22 : Barrer le(s) code(s) syntaxiquement incorrect(s).

```
typedef struct _cell {
       int val;
       _cell *suiv;
} cell;

cell c;
```

```
typedef struct _cell {
       int val;
       struct _cell *suiv;
} cell;

cell c;
```

```
struct cell {
       int val;
       struct cell *suiv;
};

struct cell c;
```

```
struct cell {
       int val;
       cell *suiv;
};

cell c;
```

Question 23 : Soit la fonction d'annulation d'un nombre *nbr* de coups dans une liste doublement chaînée des coups *l*, il manque UNE ligne très importante dans cette fonction, quelle est cette ligne ?

```
void annuler_coups(liste_t *l, unsigned nbr)
{
      unsigned i;
      cellule_t *tmp;
      for (i = 0; i < nbr; i ++) {
            tmp = l->queue;
            l->queue = l->queue->prec;
            l->queue->suiv = NULL;
      }
      l->nbr -= nbr;
}
```

Question 24 : Corriger la syntaxe de ce code :

```
int []fct(int tab[])
{
      return tab;
}
```

Question 25 : Quelle valeur vaut *x* après l'exécution de ce code ?

```
int x;
int *p;
p = &x;
x = 5;
(*p) ++;
```

5 6 Une autre valeur Erreur à la compilation ou à l'exécution

Question 26 : En supposant que le tableau tab est à l'adresse 12345678, qu'affiche le code ci-dessous ?

```
int32_t tab[] = {0, 1, 2, 3};
int32_t *p = tab + 2;
printf("%u\n", (unsigned) p);
```

12345678 12345680 12345682 12345684 12345686

Question 27 : Justifier pourquoi le code ci-dessous n'est pas correct.

```
char *fct()
{
      char chaine[5];
      chaine[0] = chaine[2] = 't';
      chaine[1] = chaine[3] = 'o';
      chaine[4] = '\0';
      return chaine;
}
```

Question 28 : Qu'affiche le code suivant ?

```
#define TAILLE 6 + 7;
printf("%u\n", TAILLE * 3);
```

39 27 Il y a une erreur de syntaxe On ne peut pas savoir

Question 29 : A quelle ligne ci-dessous est équivalente l'instruction l->tete (avec l un pointeur sur une structure contenant le champ tete) ?

`*l.tete` `*(l.tete)` `(*l).tete` Aucune

Question 30 : Qu'affiche le code suivant ?

```
int *p = malloc(sizeof(int));
printf("%d\n", *p);
```

0 NULL L'adresse de p On ne peut pas savoir

Question 31 : Qu'affiche le code suivant en supposant que l'utilisateur entre 5 au clavier ?

```
int x;
scanf("%d", &x);
if (x = 4) printf("Un");
else if (x = 6) printf("Deux");
else printf("Trois");
```

Un Deux Trois On ne peut pas savoir

Question 32 : Soit la procédure :

```
void f(int x) {
x = x+1;
}
```

Qu'affiche le code suivant ?

```
int x = 42;
f(x);
printf("%d\n",x);
```

41 42 43 L'adresse de x

Question 33. La libc contient une fonction qsort permettant de trier un tableau. Sa page de man (légèrement simplifiée ici) est la suivante :

Nom :
qsort - Trier une table.

Synopsis :

```
#include <stdlib.h>

void qsort(void *base, int32_t nmemb, int32_t size, int(*compar)(const void *, const
void *));
```

Description :

La fonction qsort() trie une table contenant nmemb éléments de taille size. L'argument base pointe sur le début de la table.

Le contenu de la table est trié en ordre croissant, en utilisant la fonction de comparaison pointée par compar, laquelle est appelée avec deux arguments pointant sur les objets à comparer.

La fonction de comparaison doit renvoyer un entier inférieur, égal, ou supérieur à zéro si le premier argument est respectivement considéré comme inférieur, égal ou supérieur au second. Si la comparaison des deux arguments renvoie une égalité (valeur de retour nulle), l'ordre des deux éléments est indéfini.

Valeur Renvoyée :

La fonction qsort() ne renvoie pas de valeur.

Compléter le programme suivant (en ajoutant si nécessaire d'autre(s) fonction(s)) pour qu'il trie le tableau par l'intermédiaire de qsort, puis l'affiche.

```
int main()
{
        int32_t tab[] = {14, 22, 13, 10, 4, 0, -7, 11};

        ...
}
```