- Développement en langage C : révision n°2 -

A une question correspond une seule réponse juste, donc cocher la bonne réponse !
Barème : une bonne réponse = +1 - pas de réponse = 0 - une mauvaise réponse = -1

Q1 . Que se passe-t-il après un **`return`** ?

☐ La fonction s'arrête et renvoie le résultat indiqué

☐ La fonction continue et renvoie le résultat indiqué

☐ La fonction continue et ne renvoie pas de résultat

Q2 . Dans quel cas l'instruction **`return`** n'est pas obligatoire ?

☐ Quand la fonction ne prend aucun paramètre en entrée

☐ Quand la fonction doit renvoyer 0

☐ Quand la fonction est de type **`void`**

Q3 . Que sont les paramètres d'une fonction ?

☐ Des indications sur le nom de la fonction

☐ Des variables qu'on lui envoie pour qu'elle puisse travailler

☐ Des indications sur la valeur qu'elle doit renvoyer

Q4 . Laquelle de ces affirmations est fausse ?

☐ Une fonction n'est pas obligée de renvoyer une valeur

☐ Une fonction peut renvoyer une valeur de n'importe quel type de variable

☐ Une fonction peut renvoyer plusieurs valeurs

Q5 . Quel est le problème de cette fonction qui est censée calculer le carré de la variable **`nombre`** ?

```
long carre(long nombre)
{
    long resultat = 0;
    resultat = nombre * nombre;
}
int main (void)
{
 long r, x = 2;
 r = carre(x);
 return 0;
}
```

☐ La fonction ne retourne aucune valeur

☐ Le paramètre **`nombre`** n'a pas été initialisé

☐ Le variable **`x`** ne porte pas le bon nom

Q6 . Qu'affiche ce programme ?

```
void f(int a, int b) {
   int c;
   c = a;    a = b;    b = c;
}
void main (void) {
 int a = 3; int b = 5;
 f(a, b);
 printf("a = %d et b = %d\n", a, b);
}
```

☐ **`a = 3 et b = 5`** ☐ **`a = 5 et b = 3`** ☐ **`a = 5 et b = 5`** ☐ il y a une erreur

Q7 . Quelle est la valeur affichée ?

```
void fonction1(int x) {
 x = x + 10; printf ("%d ", x);
}
int fonction2(int y) {
 return y + 20;
}
void main (void) {
 int x = 1;
 fonction1(x);  printf ("%d ", x);
 x = fonction2(x);  printf ("%d\n", x);
}
```

☐ 11 11 31 ☐ 11 1 31 ☐ 11 11 21 ☐ 11 1 21 ☐ c'est une erreur

Q8 . Quelle est la valeur affichée ?

```
void main(void) {
 int t[5] = { 1, 2, 3, 4, 5 };  int somme = 0;
 for(i = 1;i < 5;i++) somme = somme + t[i];
 printf("%d\n", somme);
}
```

☐ 15 ☐ 14 ☐ 10 ☐ il y a une erreur

Q9 . Quelle est la valeur affichée ?

```
void fonction(int a[]) {
  a[1] = 10;
}
void main(void) {
 int T[] = {1, 2, 3};
 fonction(T); printf("%d", T[1]);
}
```

☐ 10 ☐ 1 ☐ 2 ☐ c'est une erreur

Q10 . Quelle est la valeur affichée ?

```
void main(void) {
 int plateau [3][3] = {  { 0, 1, 0 }, { 1, 0, 1 }, { 0, 1, 0 } };
 printf("plateau[1][2] = %d\n", plateau[1][2]);
}
```

☐ 1 ☐ 0 ☐ il y a une erreur

Q11 . Quelle est la valeur affichée ?

```
int estTrie(int t[], int nb) {
        for(int i=0;i<nb;i++)
                if(t[i-1] > t[i])
                        return 0;
        return 1;
}
int main(void)
{
 int t[5] = { 1, 2, 3, 4, 5 };

 printf("%d\n", estTrie(t, 5));
 return 0;
}
```

☐ 0 ☐ 1 ☐ provoque une erreur

Q12 . Quelle est la valeur affichée ?

```
#define MAX 10
int f(int n, int t[MAX]) {
   int l, c;
   l = 0;
   while(n != 0) {
      c = n % 2; n = n / 2; t[l++] = c;
   }
   return l;
}
void main()
{
   int tab[MAX]; int i, l, x;

   x = 12;
   l = f(x, tab);
   for (i=l-1; i>=0; i--)
      printf("%i", tab[i]);
   printf("\n");
}
```

☐ 0011 ☐ 1100 ☐ 1100000000 ☐ 6310 ☐ provoque une erreur

Q13 . Dans le programme précédent, quelle est la valeur de **`l`** à la fin de l'exécution ?

☐ 0 ☐ 4 ☐ **`MAX`** ☐ 5 ☐ rien

Q14 . Dans le source de Q12, à partir de quelle valeur de **`x`** le programme risque de provoquer une erreur ?

☐ 1023 ☐ 1024 ☐ 256 ☐ -12 ☐ il n'y a pas de risque

Conseil : reprendre le source de Q12 et renommer les variables et la fonction f pour leur donner un sens puis définissez ce que fait le programme. Pour finir, proposer une sécurisation de ce programme afin qu'il ne provoque pas d'erreur de segmentation.

Q15 . Qu'affiche ce programme ?

```
#define N   5 //nombre de lignes de la grille
#define M   5 //nombre de colonnes de la grille
void foo(char grille[N][M], char mot[]) {
       int n, m;
       for(m=0;m<M;m++) {
              for(n=0;n<N;n++)
                     mot[n] = grille[n][m];
              mot[n] = 0x00; //fin de chaine
       }
}
void main()
{
       char grille[N][M] = {        {'B', 'A', 'B', 'A', 'R'},
                                    {'A', 'R', 'T', 'R', 'A'},
                                    {'Z', 'A', 'M', 'A', 'T'},
                                    {'A', 'T', 'Y', 'N', 'E'},
                                    {'R', 'A', 'S', 'E', 'R'} } ;
       char mot[M+1];
       foo(grille, mot);
       printf("%s\n", mot);
}
```

☐ **`BABAR`** ☐ **`BAZAR`** ☐ **`RASER`** ☐ **`RATER`** ☐ aucun des quatre