Exercices programmation – Révision C

Les exercices ci-dessous sont principalement basés sur les erreurs faites lors des tests et projets des années précédentes. Vous êtes encouragé à tester les codes ci-dessous sur votre machine. Si vous avez des doutes ou des questions n'hésitez pas à indiquer le numéro de la question sur le forum Ed.

Q1. Que retourne `F1()`? Pourquoi ? idem si k est défini comme un `short`? et `char`?
Est-ce possible de faire `return k` au lieu de a ? Pourquoi ?

```
int F1(void){
    int a=0;
    for (int k=1;k>0;k++) {a=k;}
    return a;
}
```

Q2. Est-ce que `F2()` compile ? Pourquoi ? Quelle est la taille de A[] ?

```
void F2(void){
    char A[];
}
```

Q3. Que se passe-t-il lors de l'exécution de `F3()`? Pourquoi ?

```
void F3(void){
    char *B;
    B[1] = 'x';
}
```

Q4. Que se passe-t-il lors de l'exécution de `F4()`? Pourquoi ?

```
void F4(void){
    char* D="hello";
    D[0] = 'y';
}
```

Q5. Que se passe-t-il lors de l'exécution de `F5()`? Pourquoi ?

```
void F5(void){
    char D[]="hello";
    D[0] = 'y';
}
```

Q6. Qu'affiche `F6()`? Pourquoi ?

```
void F6(void){
    char D[]="hello";
    D[5] = 'y';
    printf("%s\n",D);
}
```

Q7. Que vaut `F`? Pourquoi ?

```
int E[]= {4,5,6};
int F = sizeof(E)/sizeof(E[0]);
```

Q8. Que vaut `e` après l'appel à `F8()`? Pourquoi ? idem si `E[]= {7};` ?

```
int F8(int E[]) {
       return sizeof(E)/sizeof(E[0]);
}

int E[]= {4,5,6};
int e = F8(E);
```

Q9. Qu'affiche `F9()`? Pourquoi ? Quel est la taille de A[] et B[] ?

```
void F9(void){
    char A[]={'x'};
    char B[]="x";
    printf("%s, %s\n",A,B);
}
```

Q10. Qu'affiche `F10()`? Pourquoi ?
Si vous êtes sur OSX/linux essayer votre code sur Windows et vice-versa, y a-t-il une différence, pourquoi ?

```
void F10(void){
    int i = 7;
    printf("%d\n", i++ * i++);
}
```

Q11. Est-ce que `F11()` compile ? Pourquoi ?

```
void F11(void){
    int* i, j;
    *j=3;
}
```

Q12. Est-ce que `F12()` compile ? Pourquoi ?
Est-ce que `F12()` s'exécute correctement ? Pourquoi ?

```
void F12(void){
    char *p;
    *p = malloc(10);
}
```

Q13. Est-ce que `F13()` compile ? Pourquoi ? Est-ce que `F13()` s'exécute correctement ? Pourquoi ? Essayez votre code sur un autre OS, avez-vous le même résultat ?

```
void F13(void){
    double a[1024][1024];
    a[0][0] = 0;
}
```

Q14. La/lesquelles de ces déclaration de `main()` est/sont correcte(s)? Pourquoi ?

```
int main(void)
int main(int argc, const char * argv[])
int main(int argc, const char * argv[], const char *envp[])
```

Q15. Quel est la valeur minimale de `argc` ? que contient `argv[0]` ?

```
int main(int argc, const char * argv[])
```

Q16. L'exécutable `myProg` est appelé avec `./myProg 42`
Que valent `argc, argv[0], argv[1], argv[2]` ?
Comment afficher le paramètre 42 ?

```
int main(int argc, const char * argv[]){

       printf("%...", argv[...]) ;
}
```

Q17. Comment convertir un `char` contenant un **chiffre** en un `int`?

```
char c='8';
int  cc = … ; // cc doit valoir 8
```

Q18. Qu'affiche `F18()`? Pourquoi ? Essayez votre code sur un autre OS, avez-vous le même résultat ? Comment réécrire le code ci-dessous pour enlever l'ambiguïté ?

```
void F18(void){
    int i = 3;
    i = i++;
    printf("i: %d\n",i);
}
```

Q19. Qu'affiche `F19()`? Pourquoi ? Essayez votre code sur un autre OS

```
int aa(void){printf("aa "); return 1;}
int bb(void){printf("bb "); return 2;}
void F19(void){
    printf("%d %d\n", aa(),bb());
}
```

Q20. `F20()` regarde si `b` se trouve entre `a` et `c`, pourquoi le résultat est incorrect?

```
int F20(int a,b,c) {
    if (a < b < c) return  1;
              else return -1;
}
```

Q21. Qu'affiche `F21(1000)`? Pourquoi ?

```
void F21(short a){
    short b= a*a;
    printf("%d x %d = %d\n", a,a,b);
}
```

Q22. Qu'affiche `F22(3)`? Pourquoi ?

```
void F22(int a){
    double b= a/2;
    printf("%d/2 = %f\n", a,b);
}
```

Q23. Qu'affiche `F23(3)`? Pourquoi ?

```
void F23(int a){
    double b= a/2.0;
    printf("%d/2.0 = %f\n", a,b);
}
```

Q24. Qu'affiche `F24(3)`? Pourquoi ?

```
void F24(double a){
    int b= a/2;
    printf("%f/2.0 = %d\n", a,b);
}
```

Q26. Qu'affiche `F25()`? Pourquoi ?

```
void F25(void){
    double a=12.34;
    int    b=456;
    printf("1) a:%f, b: %d\n", a,b);
    printf("2) a:%d, b: %f\n", a,b);
    printf("3) a:%d, b: %d\n", a,b);
    printf("4) a:%f, b: %f\n", a,b);
    printf("5) a:%d, b: %d\n", (int)a,b);
    printf("6) a:%d, b: %d\n", (double)a,(double)b);
    printf("7) a:%f, b: %f\n", (double)a,(double)b);
}
```

Q26. Est-ce `F26(100)` s'exécute correctement sur votre machine?
Et pour `F26(10000000)`? Pourquoi ?
Comment modifier `F26()`pour qu'elle s'exécute pour toutes les valeurs de sz ?

```
void F26(unsigned int sz){
    int a[sz];
    a[0]=2;
}
```

Q27. Quelle est la différence entre les 2 lignes ci-dessous :

```
const int MAXSIZE = 100;
#define MAXSIZE 100
```

Q28. Qu'affiche `F28()`? Pourquoi ?

```
void F28(void){
    char a[5]="hello";
    printf("%s\n",a);
}
```

Q29. Qu'affiche `F29()`? Pourquoi ?

```
void F29(void){
    printf("%.1f, %.20f\n",3.1, 3.1);
}
```

Q30. Combien de fois s'exécute la boucle dans `F30()`? Pourquoi ?

```
void F30(void){
    for (double x=0.0; x != 10.0; x += 0.1)
       printf("x:%f\n",x);
}
```

Q31. Qu'affiche `F31(5)`? Pourquoi ?

```
int F31(int a){
    if (a>2)
        printf("a>2\n");
    if (a>5) ;
        printf("a>5\n");
    return a;
    printf("a: %d\n",a);

}
```

Q32. Sur ma machine `F32()` affiche '8', mais sur la VM elle affiche '4' ? Pourquoi ?

```
void F32(void){
    int *p;
    printf("sizeof(p): %ld\n", sizeof(p));
}
```