Partie I

QCM : Pour chacune des questions ci-dessous, indiquez sur votre copie la bonne réponse.
Points : 3

1) L'expression « `e /= 2 ;` » :
   a) contient une erreur de syntaxe
   b) équivaut à l'expression « `e = e / 2 ;` »
   c) calcule le carré de `e` (e^2)
   d) aucune des réponses précédentes n'est correcte

2) L'expression « `e *= e ;` » :
   a) contient une erreur de syntaxe
   b) équivaut à l'expression « `e = e * 2 ;` »
   c) calcule le carré de e (e^2)
   d) aucune des réponses précédentes n'est correcte

3) Que fait l'expression : « `if (e%2 != 0) p*=y;` » ?
   a) Rien, elle contient une erreur de syntaxe
   b) Elle multiplie la valeur de `p` par celle de `y`
   c) Elle multiplie la valeur de `p` par celle de `y` uniquement si `e` est impair
   d) Elle multiplie la valeur de `p` par celle de `y` uniquement si `e` est pair
   e) Aucune des réponses précédentes n'est correcte

4) Le programme

```
int main() {
      int i=0 ; while(i<10) printf("%d ",i); i++;
}
```

   a) va afficher 9 nombres
   b) va afficher 11 nombres
   c) va afficher 10 nombres
   d) va boucler indéfiniment
   e) contient une erreur de syntaxe
   f) Aucune des réponses précédentes n'est correcte

5) Le programme

```
int main() {
      int i=10; do printf ("%d ", i--); while (i>=0);  }
```

   a) va afficher 9 nombres
   b) va afficher 11 nombres
   c) va afficher 10 nombres
   d) va boucler indéfiniment
   e) contient une erreur de syntaxe
   f) Aucune des réponses précédentes n'est correcte

6) Les instructions suivantes « `if (a<5) { printf("Bonjour") ; a=a+1 ; }` »
   a) affichent « Bonjour » et augmentent la valeur de `a` quelque soit valeur la valeur de `a`
   b) n'affichent pas « Bonjour », ni augmentent la valeur de `a` quelque soit la valeur de `a`
   c) affichent « Bonjour » quelque soit la valeur de `a`
   d) affichent « Bonjour » et augmentent la valeur de `a` uniquement si `a` est inférieur à 5
   e) n'affichent pas « Bonjour », ni augmentent la valeur de `a` si `a` est inférieur à 5
   f) Aucune des réponses précédentes n'est correcte

7) Les instructions suivantes « `if (h>5 && h<=18) printf("Bonjour") ; else printf ("Bonsoir") ; }` »
   a) affichent toujours « Bonjour »
   b) affichent toujours « Bonsoir »
   c) affichent « Bonjour » si la valeur de `h` est égale à 5
   d) affichent « Bonsoir » uniquement si `h` est inférieur à 5 ou si `h` est supérieur ou égale à 18
   e) Aucune des réponses précédentes n'est correcte

8) L'instruction « `if (a%2 == 0) printf("bonjour") ;` »
   a) contient une erreur de syntaxe
   b) n'affiche rien, quelque soit la valeur de `a`
   c) affiche « bonjour » quand `a` est un entier impair
   d) affiche « bonjour » quand `a` est un entier pair
   e) Aucune des réponses précédentes n'est correcte

9) L'opérateur == permet en langage C de :
   a) réaliser une affectation
   b) tester une égalité
   c) comparer deux variables
   d) convertir un `int` en `float`
   e) Aucune des réponses précédentes n'est correcte

10) Le programme « `int main(){ int b=5, x=12, z ; z = (2+5*x+4)/b-3 ; printf ("%d",z); }` » affiche la valeur :
   a) 10
   b) 10.2
   c) 17
   d) 17.6
   e) Aucune des réponses précédentes n'est correcte

11) Quelle sera la valeur de `j` après l'exécution du bloc d'instructions suivant :

```
j = 0;
switch (i) {
   case 3:
        j++;
   case 2:
        j+=2;
   case 1:
        j=3+;
}
```

a) 3 si i=3, 2 si i=2, 1 si i=1
b) 1 si i=3, 2 si i=2, 1 si i=3
c) 6 si i=3, 5 si i=2, 3 si i=1
d) 6 quelque soit la valeur de i
e) Aucune des réponses précédentes n'est correcte

12) L'instruction « `if (size=0) i++ ; else i--;` »

a) contient une erreur
b) incrémente la variable i si size est positif
c) décrémente la variable i si size est négatif
d) incrémente la variable i si size est différent de 0
e) décrémente la variable i si size est différent de 0

13) Le programme

```
int main () {
   int i;
   for (i = 0; i < 10; i++);
   printf ("%d\n", i);
}
```

a) va afficher 1 nombre
b) va afficher 9 nombres
c) va afficher 10 nombres
d) va afficher 11 nombres
e) Aucune des réponses précédentes n'est correcte

14) Le programme

```
int main ()  {
   int ASCII;
   ascii = getchar() ;
  printf ("%d\n", ascii);
}
```

a) contient une erreur
b) va afficher le code ASCII du caractère fourni par l'utilisateur
c) va afficher le caractère fourni par l'utilisateur
d) Aucune des réponses précédentes n'est correcte

15) Le programme

```
int main () {
   int i;
   for (i = 0; i < 10; i++)
      printf ("%d\n", i);
}
```

a) va afficher 1 nombre
b) va afficher 9 nombres
c) va afficher 10 nombres
d) va afficher 11 nombres
e) Aucune des réponses précédentes n'est correcte

Partie II

Questions à lacunes : Pour chacune des questions ci-dessous, remplissez les lacunes soulignées dans le programme (indiquez sur votre copie uniquement les mots/expressions qui remplissent les lacunes de chaque question dans l'ordre).
Points : 3

16) Remplissez les lacunes soulignées dans le programme ci-dessous :

```
#include <stdio.h>

int main ()
{
   float x,y,t;

   printf ("Entrez x : ");
   _______ ("___", &x);
   printf ("Entrez y : ");
   _______ ("___", &y);

   if (x<y) {
      t = ____;
      x = y;
      y = ____;
   }
   printf ("Ecart : %f", (x-y));
}
```

17) Sachant que le programme ci-dessous doit afficher les diviseurs d'un nombre entier fourni par l'utilisateur, remplissez les lacunes indiquées dans le programme :
Attention : Les diviseurs d'un nombre entier positif n sont tous les nombres entiers positifs i inférieurs à n (0< i ≤ n) pour lesquels la division n÷ i est exacte.

```
#include <stdio.h>

int main () {
   int _____, _____;

   printf ("Entrez n : ");
   scanf ("%d", &n);

   i = _____;

   printf ("Diviseurs de %d : ", n);
   ______ (i>0) {
      if ((_______) == 0) {
         printf (" %d ", i);
      }
      i--;
   }
}
```

18) Remplissez les lacunes soulignées dans le programme ci-dessous, sachant que celui-ci affiche un tableau de conversion entre les unités Km et Milles (1Mille = 1,61 Km).

```
#include <stdio.h>

int main () {
    ______ km;
    int mi;

    _____ ("Milles \t  Km\n");

    _____ (mi = 1; mi < 50; mi++) {
        km = 1.62 * (float) mi;
        ______ ("%d  \t  %f \n", mi, km);
    }

}
```

Partie III

Conversion : Pour chacune des questions ci-dessous, vous devez réécrire le programme fourni, en utilisant pour cela l'instruction de contrôle qui vous est demandée.
Points : 3

19) Réécrire le programme ci-dessous en utilisant une instruction de type *if…else* à la place de l'instruction *switch…case*.

```
#include <stdio.h>
int main () {
    int nb_blancs, nb_0_1, nb_a_e, nb_p_m, nb_autres;
    char c;

    nb_blancs = 0;
    nb_autres = 0;
    nb_0_1 = 0;
    nb_a_e = 0;
    nb_p_m = 0;

    printf ("Entrer autant des caracteres que vous voulez.\n");
    printf ("Tapez S pour sortir\n");

    c = getchar();
    while (c != 'S' && c != 's') {  //la lettre S fait sortir
        switch (c) {
          case '0':
          case '1':
                    nb_0_1++;
                    break;
          case 'a':
          case 'e':
          case 'A':
          case 'E':
                   nb_a_e++;
                   break;
          case 'p':
          case 'P':
          case 'M':
          case 'm':
                    nb_p_m++;
                    break;
           case ' ':
           case '\n':
           case '\t':
                     nb_blancs++;
                     break;
          default:
                     nb_autres++;
        } //switch

        c = getchar();
    } //while
```

```
    printf ("\n Resume: %d 0/1  %d a/e (gauche) %d p/m (droite) %d
blancs  %d autres \n", nb_0_1, nb_a_e, nb_p_m, nb_blancs, nb_autres);
}
```

20) Sachant que le programme ci-dessous calcule le factoriel d'un numéro *n* fourni par l'utilisateur (*n!*), réécrivez le programme en utilisant une boucle *for* à la place de la boucle *do…while*.
Attention : le factoriel d'un entier positif n est défini comme n! = 1 × 2 × … × n-1 × n.

```
#include <stdio.h>
int main () {
   int n=0, fact = 1;
   int copie;

   printf ("Entrez n : ");
   scanf ("%d", &n);
   copie = n;

   do {
      fact *= n;
      n--;
   } while (n>0);

   printf ("%d ! = %d", copie, fact);
}
```

21) Le programme ci-dessous calcule la somme des N premiers termes de la série harmonique *1 + 1/2 + 1/3 + … + 1/N*. Réécrivez le programme en utilisant une boucle *do…while* à la place de la boucle *while*.

```
#include <stdio.h>
int main() {
  int N=10;
  int i;
  int som;
  float harmo;

  harmo = 0.0;
  som = 0; i = 1; /* initialisation des variables */

  while (i<=N){
    som = som + i;
    harmo = harmo + 1/(float)i;
    i++;
  }

  printf("La somme des %d premiers entiers est : %d\n", N, som);
  printf("La somme des %d premiers termes de la serie harmonique vaut
: %f\n", N,harmo);
}
```

Partie IV

Traces : Pour chacune des questions ci-dessous, réalisez la trace d'exécution des programmes indiqués.
Points : 3

22) Faites le trace d'exécution du programme suivant.

```
#include <stdio.h>
main() {
    int p, s, i, t, n;
    n=6;
    p=0; // Valeur de u1
    s=0; // Valeur de u2
    // Point d'observation 1
    for(i=3; i<=n; i++){
        t = s;
        s = 2*p+s+1;
        p = t; // Point d'observation 2
    }
    printf("La valeur de u%i est : %i\n", i-1, s);
    // Point d'observation 3
}
```

23) Tracez l'exécution du programme suivant.

```
#include<stdio.h>
main(){
int min, max;
   int reste;
   int i, n;
   min = 2;
   max = 7;
   printf ("Les premiers entre %d et %d sont : ", min, max);
   n = min;
   //pont d'observation 1
   while (n<=max) {
       i=2;
       reste = 1;
       //pont d'observation 2
       do {
           if (n != i)
              reste = n%i;
           i++;
           //pont d'observation 3
       } while (i<=n && reste != 0);
       //pont d'observation 4
       if (reste != 0) {
           printf (" %d ", n);
       }
       n += 1; //pont d'observation 5
   }
  //pont d'observation 6
}
```

24) Réalisez la trace d'exécution du programme ci-dessous, en supposant que l'utilisateur a fourni les informations suivantes (dans l'ordre) : « 500 », « 100 », « 0.01 ».

```
#include<stdio.h>
int main(){
  int i;
  float S,Sact,Rembparmois,interet;
  float tauxmensuel,totalremb;

  //pont d'observation 1
  printf("Somme empruntee ? ");
  scanf("%f",&S);
  printf("Remboursements mensuels souhaites ? ");
  scanf("%f",&Rembparmois);
  printf("Taux mensuel de l'emprunt (0.01 par exemple) ?");
  scanf("%f",&tauxmensuel);
  printf("\n");

  //pont d'observation 2
  if (Rembparmois<=S*tauxmensuel)
    printf("IMPOSSIBLE : remboursement trop faible\n");
  else
  { Sact=S;
    i=0;
    totalremb=0;
    printf("duree      Paiement     interet   capital restant\n");
    //pont d'observation 3
    do
    { i=i+1;
      interet=Sact*tauxmensuel;
      //pont d'observation 4
      if (Rembparmois > Sact+interet)
      {  Rembparmois=Sact+interet;
         Sact=0;
      }
      else
      {
        Sact -= (Rembparmois-interet);
      }
      totalremb += Rembparmois;
      printf("%4d %14.2f %10.2f %13.2f\n",
                i,Rembparmois,interet,Sact);
      //pont d'observation 5
    } while (Sact>0);
    printf("\nle montant total des remboursements est : %14.2f\n",
            totalremb);
  }
  //pont d'observation 6
}
```

Partie V

Pour chacune des questions ci-dessous, écrivez le programme pour le problème indiqué, suivant les indications données.
Points : 3

25) Un nombre est parfait quand il est égal à la somme de ses diviseurs. Par exemple :

6	=	1 + 2 + 3	est un nombre parfait.
28	=	1 + 2 + 4 + 7 + 14	est aussi un nombre parfait.
8	≠	1 + 2 + 4	n'est pas un nombre parfait.

Rappel : les diviseurs d'un nombre entier positif n sont de tous les nombres entiers compris entre 1 et n-1 qui divisent n.

Ecrire un programme qui demande à l'utilisateur un numéro entier *n,* puis qui décide et affiche si *n* est parfait ou non.

26) La compagnie d'assurance « MegaAssur » calcule le bonus/malus pour ses conducteurs de la manière suivante : le bonus/malus de l'année (cbm) correspond à celui de l'année précédente (cbmprec) multiplié par un coefficient (coef). Ce coefficient est calculé en fonction du nombre d'accidents : à chaque accident, on ajoute 0,05 à un coefficient minimum de 1 (coeff = 1 + 0,05 × nbacc). En revanche, si le conducteur n'a pas eu d'accident, il aura droit à un coefficient de 0,95. Par ailleurs, la valeur du bonus/malus est limitée entre 0,5 (valeur minimale) et 3,5 (valeur maximale). Ainsi, si le bonus/malus calculé pour un conducteur dépasse ces seuils, son bonus/malus sera de 0,5 (si la valeur calculée est inférieur à la valeur minimale) ou de 3,5 (si la valeur calculée est supérieur à la valeur maximale).

Ecrire un programme qui calcule le bonus/malus pour les conducteurs de la compagnie « MegaAssur ». Ce programme doit demander à l'utilisateur le bonus/malus correspondant à l'année précédente (cbmprec), ainsi que le nombre d'accidents (nbacc), et à partir de ces données calculer le bonus/malus pour l'année courante.