CORRECTION DES TD/TP DE PROGRAMMATION IMPÉRATIVE

Table des matières

TD n°1

I - Exercice 1

```
/*programme de calcul de circonférence*/

#include <stdio.h>

float   circonference(float rayon)
   {
      float pi;
      float circon;
      pi = 3.1415;
      circon = 2 * pi * rayon;
      return (circon);
   }

int     main(void)
   {
      float r;
      int i;
      int n;
      n = 5;
      /* saisie par l'utilisateur de n rayons et calcul des n curconferences correspondantes */
      for (i = 0; i < n; i++)
         {
            printf("Saisie du rayon n° %d \n", i);
            scanf ("%f", &r);
            printf ("La circonference est de : %.2f\n", circonference(r));
         }
      return (1);
   }
```

I - Exercice 2

```
#include <stdio.h>

int main(void)
  {
     int nombre;

     nombre = 4;
     printf("Bonjour\n");
     printf("au revoir \n"); //réalise l'affichage d'au revoir à l'écran
     return (1);
  }
```

II – Exercice 1

float r, r1, r2;	Correct
real x,y,z;	Incorrect : type inexistant
int for,main;	Incorrect : identificateurs réservés par le langage
char rs-232;	Incorrect : tirets interdits
double d1; d2; d3;	Incorrect : points virgule au lieu de virgules
long int _____3;	Correct
unsigned char c='c';	Correct

II – Exercice 2

compteur [0 .. 300] = unsigned int OU unsigned short
x, y [-120, 100] = char
mesure [-10 .. 104] = short OU int
surface [0.5 .. 150075] = float
nb1 [-1.. 1024] = short OU int
nb2 [0 .. 70000] = unsigned long
trouve [vrai, faux] = short

II – Exercice 3

1) Char
2) Short / int
3) Unsigned short / unsigned int
4) Float
5) Int
6) Float
7) long

TP n°1

Exercice 1

```
#include <stdio.h>
int main()
{
   printf("Ceci est mon premier programme");
   return 0;
}
```

Exercice 2

a)

```
#include <stdio.h>
int main()
{
   printf("La somme de 10 et 5 est %d", 10+5);
   printf("La multiplication de 10 par 5 est %d", 10*5);
   printf("La division de 20 par 6 est %.3f", 20/6);
   printf("10+5 = %d \t 10*5 = %d \t 20.0/6.0 = %.2f", 10+5, 10*5, 20./6);
   return 0;
}
```

b)

```
#include <stdio.h>
int main()
{
   int a = 10, b = 5;
   float c = 20, d = 6;
   printf("La somme de %d et %d est %d", a, b, a+b);
   printf("La multiplication de %d par %d est %d", a, b, a*b);
   printf("La division de %.0f par %.0f est %.3f", c, d, c/d);
   printf("%d+%d = %d \t %d*%d = %d \t %f/%f = %.2f", a, b, a+b, a, b, a*b, c, d, c/d);
   return 0;
}
```

Exercice 3

```
#include <stdio.h>

int main(void)
{
	int j,m,a,donnees;
	char tmp;

	printf("Introduisez la date (jour mois année) : ");
	donnees = scanf("%d%c%d%c%d", &j, &tmp, &m, &tmp, &a);
	printf("Données correctement lues : %d\n", donnees - 2);
	printf("/*****Affichage de la date saisie ****/\n");
	printf("Jour : %d\nMois : %d\nAnnée : %d\n", j,m,a);
	return (0);
}
```

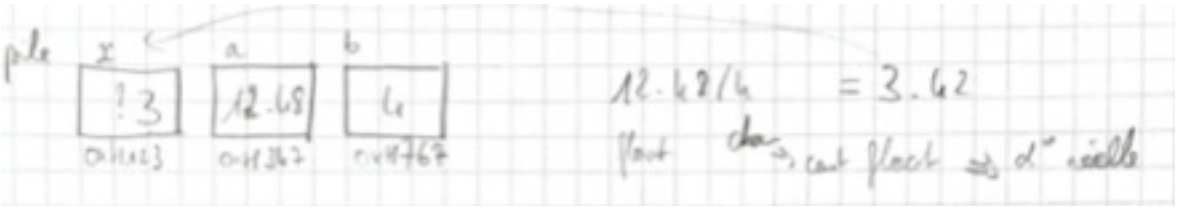
Exercice 4

```
#include <stdio.h>

int main(void)
{
	float a,b,s,p,c1,c2;

	printf("entrez deux entiers A et B : ");
	scanf("%f%f", &a,&b);
	s = a+b;
	p = a*b;
	c1 = a*a;
	c2 = b*b;
	printf("%.0f + %.0f = %.0f\n%.0f * %.0f = %.0f\ncarré de %.0f = %.0f\ncarré de %.0f =
	%.0f\n", a,b,s,a,b,p,a,c1,b,c2);
	return (0);
}
```

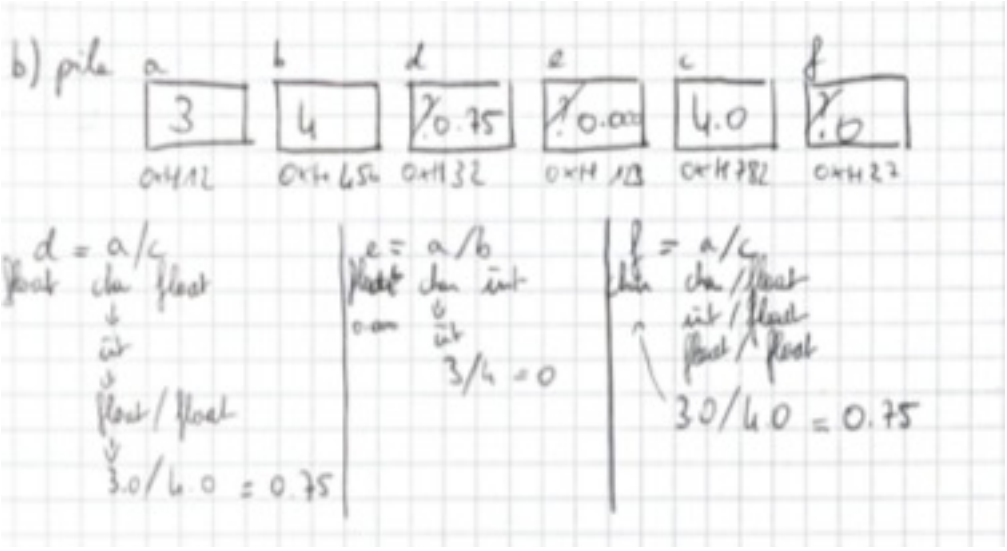
TD n°2

Exercice 1

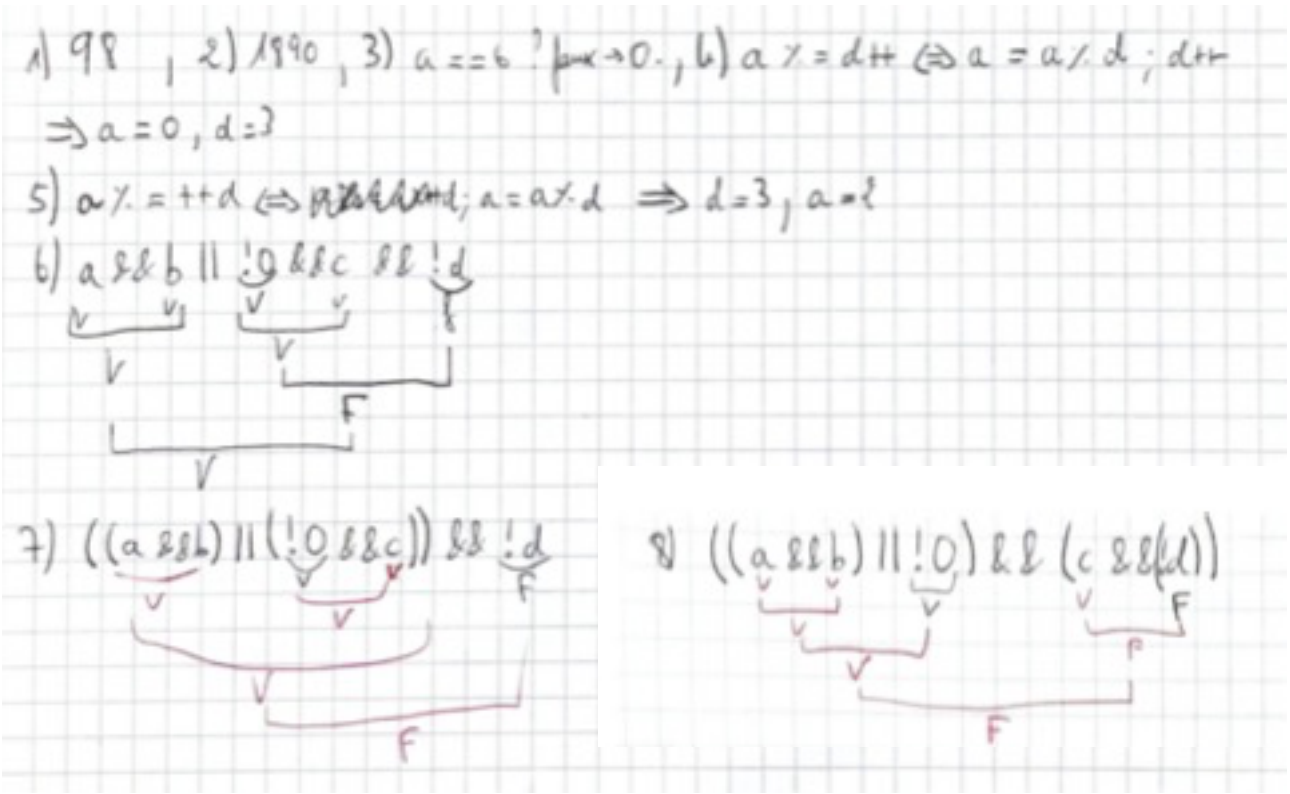
a)



b)



Exercice 2



Exercice 3

(x >= 3 && X <= 6) || (x >= 7 && x < 10)

Exercice 4

1)

```
if (a > b)
   if (a > 10)
      printf("1er choix\n");
   else if (b < 10)
      printf("2e choix\n");
   else if (a == b)
      printf("3e choix\n");
   else printf("4e choix\n");
```

2) 3)

1er choix = a > b && a > 10

2e choix = a > b && a <= 10 && b < 10

3e choix = a > b && a <= 10 && b >= 10 && a == b (IMPOSSIBLE, 3e choix ne s'affiche jamais)

4e choix = a > b && a <= 10 && b >= 10 && a != b (IMPOSSIBLE, 4e choix ne s'affiche jamais)

Exercice 5

2) On ne met jamais de variable après case

3) aucune (mais comme il n'y a pas de break, tout ce qui se trouve après le cas où on va sera affiché)

Exercice 6

0 : Nul Petit
1 : Petit
4 : Moyen Grand
10 : Grand
-5 : Grand

TP n°2

Exercice 1

```
#include <stdio.h>

/* Version exo1 a
  int main(void)
  {
  int a,b,c,d,e;

  printf("entrez 4 entiers : ");
  fflush(stdin);
  scanf("%d", &a);
  scanf("%d", &b);
  scanf("%d", &c);
  scanf("%d", &d);
  e = a+b+c+d;
  printf("Somme = %d\n", e);
  return (0);
  }*/

// Version exo1 b
int main(void)
{
   int tmp, result=0, i=0;

   printf("entrez 4 entiers : ");
   while (i < 4)
      {
         scanf("%d", &tmp);
         result += tmp;
         i++;
      }
   printf("Somme = %d\n", result);
   return (0);
}
```

Exercice 2

```
#include <stdio.h>

int main()
{
   int var1, var2;
   printf("Entrez votre division (format entier/entier) : ");
   fflush(stdout);
   scanf("%d/%d", &var1, &var2);
   printf("Le quotient vaut %d, le reste vaut %d et le quotient rationnel vaut %.2f.\n",
var1/var2, var1%var2, ((float)var1)/((float)var2));
   return (0);
}
```

Exercice 3

```
#include <stdio.h>

#define PI 3.1415

int main (void)
{
   float c, rayon, aire_c, aire_n, circonf;

   printf("cote du carre ? ");
   scanf("%f", &c);
   aire_c = c*c;
   printf("aire carre = %.2f\n", aire_c);
   rayon = c/2;
   printf("rayon = %.2f\n", rayon);
   circonf = PI*rayon*rayon;
   printf("circonf = %.2f\n", circonf);
   aire_n = aire_c - circonf;
   printf("aire surface noircie = %.2f", aire_n);
   return (0);
}
```

Exercice 4 (amélioré)

```c
#include <stdio.h>

int main()
{
   float somme;
   printf("Somme ? ");
   scanf("%f", &somme);
   somme*=100; // on multiplie par 100 pour pouvoir calculer au centime près sans avoir de virgule
   if (!((int)somme/50000 == 0)) //à chaque fois on vérifie que le rang n'est pas égal à 0 pour faire + propre
      printf("billets de 500€ : %d\n", (int)somme/50000);
   somme=(int)somme%50000;
   if (!((int)somme/10000 == 0))
      printf("billets de 100€ : %d\n", (int)somme/10000);
   somme=(int)somme%10000;
   if (!((int)somme/2000 == 0))
      printf("billets de 20€ : %d\n", (int)somme/2000);
   somme=(int)somme%2000;
   if (!((int)somme/1000 == 0))
      printf("billets de 10€ : %d\n", (int)somme/1000);
   somme=(int)somme%1000;
   if (!((int)somme/500 == 0))
      printf("billets de 5€ : %d\n", (int)somme/500);
   somme=(int)somme%500;
   if (!((int)somme/200 == 0))
      printf("pieces de 2€ : %d\n", (int)somme/200);
   somme=(int)somme%200;
   if (!((int)somme/100 == 0))
      printf("pieces de 1€ : %d\n", (int)somme/100);
   somme=(int)somme%100;
   if (!((int)somme/50 == 0))
      printf("pieces de 50cts : %d\n", (int)somme/50);
   somme=(int)somme%50;
   if (!((int)somme/20 == 0))
      printf("pieces de 20cts : %d\n", (int)somme/20);
   somme=(int)somme%20;
   if (!((int)somme/10 == 0))
      printf("pieces de 10cts : %d\n", (int)somme/10);
   somme=(int)somme%10;
   if (!((int)somme/5 == 0))
      printf("pieces de 5cts : %d\n", (int)somme/5);
   somme=(int)somme%5;
   if (!((int)somme/2 == 0))
      printf("pieces de 2cts : %d\n", (int)somme/2);
   somme=(int)somme%2;
   if (!((int)somme/1 == 0))
      printf("pieces de 1cts : %d\n", (int)somme/1);
   return 0;
}
```

Exercice 5

Explication


```
#include <stdio.h>

int main(void)
{
   int x,p,n,nb,i;

   printf("x ? ");
   scanf("%d", &x);
   printf("p ? ");
   scanf("%d", &p);
   printf("n ? ");
   scanf("%d", &n);
   //génération du masque (= 2^n - 1 << p - 1)
   nb = 1;
   for(i=0; i < n; i++)
      nb *= 2; //on trouve la puissance de 2 maximale pour le x choisit
   nb -= 1; // on soustrait 1 (formule)
   nb = nb << (p-1); // on décale à gauche p-1 bits dans nb
   nb = nb ^ x; // ou exclusif qui permet de mettre à 1 si les bits sont identiques et 0 sinon
   printf("nombre de depart = %d, nombre apres inversion des %d bits au rang %d = %d",
x,n,p,nb);
   return (0);
}
```

Exercice 6

```
#include <stdio.h>
#include <math.h>

int main()
{
	float a, b, c, des, racine_1, racine_2;

	do
	{
		printf("Entrez votre équation du second degrès :\na = ");
		fflush(stdout);
		scanf("%f", &a);
	} while (a==0);

	printf("b = ");
	fflush(stdout);
	scanf("%f", &b);
	printf("c = ");
	fflush(stdout);
	scanf("%f", &c);

	des = b*b - 4*a*c;
	if (des < 0)
	{
		printf("Pas de racines réelles pour l'équation.\n");
	}
	else if (des == 0)
	{
		racine_1 = -(b)/(2*a);
		printf("Une racine double : %.2f.\n", racine_1);
	}
	else
	{
		racine_1 = (-(b)-sqrt(des))/(2*a);
		racine_2 = (-(b)+sqrt(des))/(2*a);
		printf("Deux racines : x1 = %.2f et x2 = %.2f.\n", racine_1, racine_2);
	}
	return (0);
}
```

Exercice 7

```
#include <stdio.h>

int main()
{
   float r1, r2;
   char choix;

   printf("Entrez votre premier réel : ");
   scanf("%f", &r1);
   printf("Entrez votre second réel : ");
   scanf("%f", &r2);
   do
      {
         printf("\nAddition  A\nSoustraction  S\nMultiplication  M\nDivision  D\nQuitter
Q\n");
         printf("Votre choix ? ");
         scanf("%*c%c", &choix);
         printf("\n");
         switch (choix)
            {
            case 'm':case 'M' : printf(" %.2f * %.2f = %.2f\n", r1, r2, r1 * r2);
               break;
            case 'a':case 'A' : printf("%.2f + %.2f = %.2f\n", r1, r2, r1 + r2);
               break;
            case 's':case 'S' : printf("%.2f - %.2f = %.2f\n", r1, r2, r1 - r2);
               break;
            case 'd':case 'D' : printf("%.2f / %.2f = %.2f\n", r1, r2, r1 / r2);
               break;
            case 'q':case 'Q' :
               break;
            default : printf("%c n'est pas un choix valide.\n", choix);
            }
      }
   while (choix!='q'&&choix!='Q');
   return(0);
}
```

Exercice 8

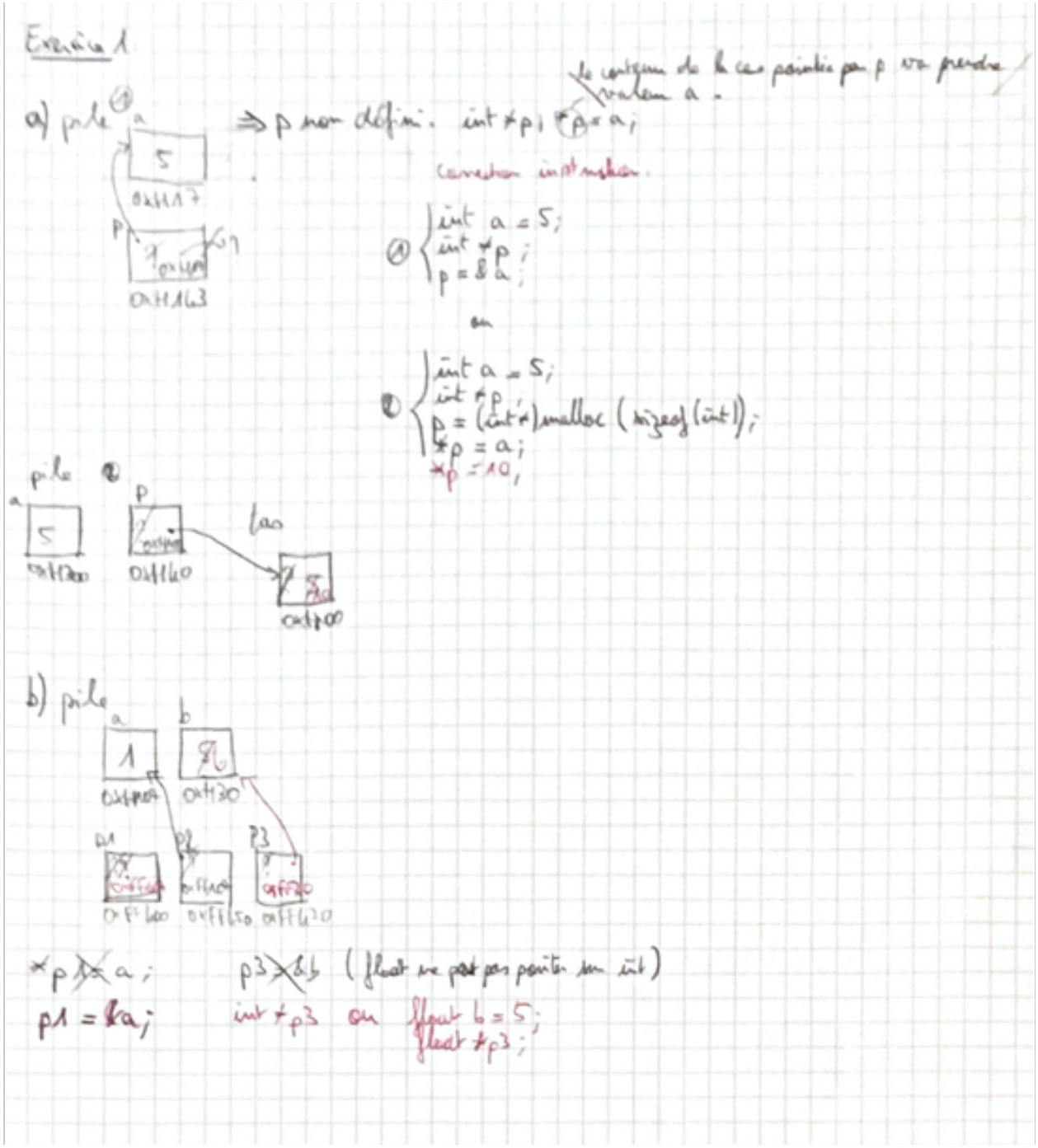
```
#include <stdio.h>

int main(void)
{
   int nb, tmp, somme=0, produit=1, moyenne, i;

   printf("Nb d'entiers désirés ? ");
   scanf("%d", &nb);
   for(i = 0; i < nb; i++)
      {
         printf("Nombre n°%d : ", i+1);
         scanf("%d", &tmp);
         somme += tmp;
         produit *= tmp;
      }
   moyenne = somme / nb;
   printf("somme = %d, produit = %d, moyenne = %d\n", somme, produit, moyenne);
   return (0);
}
```

Exercice 9

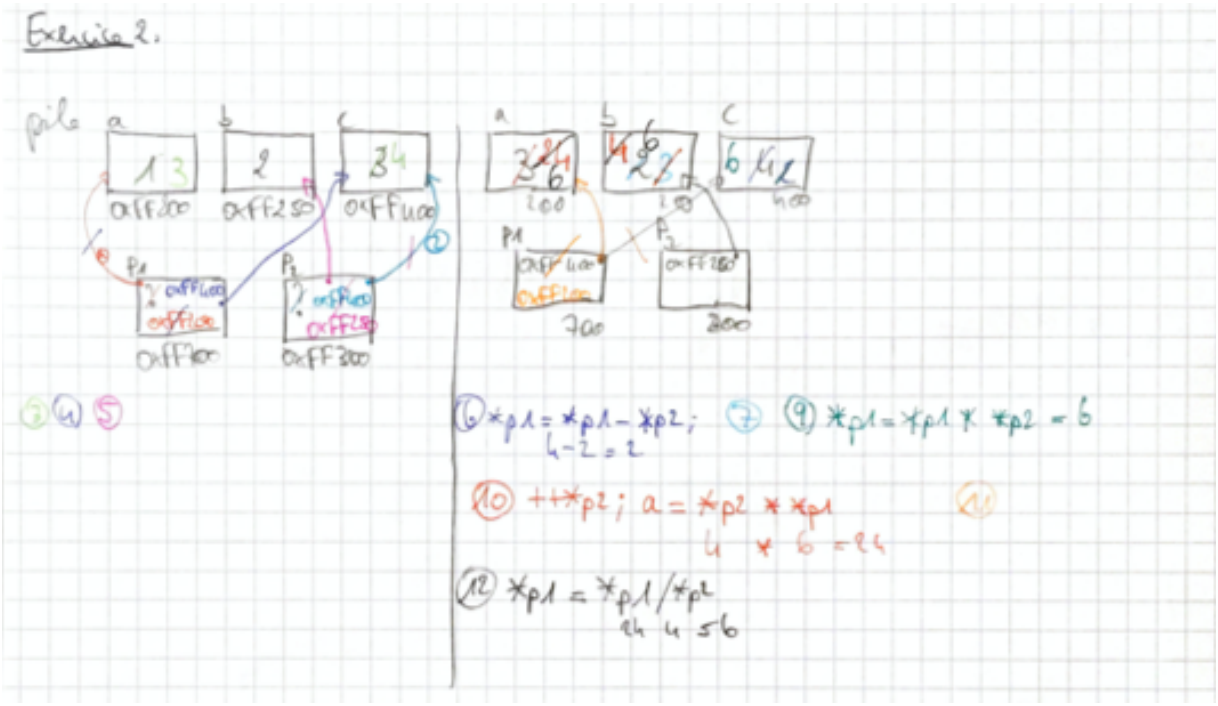
```
#include <stdio.h>

int main(void)
{
   int a,b,c;

   printf("a = ");
   scanf("%d", &a);
   printf("b = ");
   scanf("%d", &b);
   printf("c = ");
   scanf("%d", &c);
   if (a == b && b == c)
      printf("triangle equilateral\n");
   else if (a == b || a == c || b == c)
      printf("triangle isocele\n");
   else if (a*a == (b*b + c*c))
      printf("triangle rectangle\n");
   else
      printf("triangle quelconque\n");
   return (0);
}
```

TD n°3

Exercice 1



Exercice 2


Exercice 2.
pile
a
b
c
0xFF300
0xFF250
0xFF100
p1
p2
0xFF100
0xFF300
0xFF100
0xFF250
A
B
C
100
150
400
p1
p2
700
200
4 5
6 *p1 = *p1 - *p2 ; 4-2 = 2
7
9 *p1 = *p1 * *p2 = 6
10 ++*p2 ; a = *p2 * *p1
4 * 6 = 24
11
12 *p1 = *p1 / *p2
24 4 = 6

TP n°3

Exercice 1

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
  int x = 15;
  printf("Après définition et initialisation de x\n");
  printf("adresse de x: %p, valeur de x: %d\n", &x,x);
  int *pInt = &x;
  printf("Après définition et initialisation de pInt\n");
  printf("adresse de pInt: %p, contenu de pInt: %p, valeur de la case pointée par pInt:
%d\n", &pInt, pInt, *pInt);
  int *ptrInt;
  if (!(ptrInt = (int *) malloc(sizeof(int))))
    printf("Erreur d'allocation mémoire.\n");
  *ptrInt = 100;
  printf("Après allocation et initialisation de ptrInt\n");
  printf("adresse de ptrInt: %p, contenu de ptrInt: %p, valeur de la case pointée par ptrInt:
%d\n", &ptrInt, ptrInt, *ptrInt);
  free(ptrInt);
  printf("****** Après instruction de modification de la valeur pointée par pInt ******\n");
  *pInt = 200;
  printf("adresse de pInt: %p, contenu de pInt: %p, valeur de la case pointée par pInt:
%d\n", &pInt, pInt, *pInt);
  printf("adresse de x: %p, valeur de x: %d\n", &x,x);
}
```

Exercice 2

```
#include <stdio.h>
#include <stdlib.h>

#define MAX 10

int main(void)
{
  int nb, i;
  float *ptrMax=NULL, *ptrMin=NULL, *ptr;
  do
    {
      printf("Nb de valeurs réelles à saisir ? ");
      scanf("%d", &nb);
    } while (nb <= 0 || nb > MAX);
  if (!(ptr = (float *) calloc(nb,sizeof(int))))
    printf("erreur d'allocation mémoire.\n");
  for(i=0; i < nb; i++)
    {
      printf("valeur réelle n°%d ? ", i+1);
      scanf("%f", &ptr[i]);
    }
  ptrMax = ptr;
  ptrMin = ptr;
  for(i = 0; i < nb; i++)
    {
      if (*ptrMax < *(ptr+i))
        ptrMax = ptr+i;
      if (*ptrMin > *(ptr+i))
        ptrMin = ptr+i;
    }
  printf("****** Affichage de la suite de valeurs réelles saisies ******\n");
  for(i = 0; i < nb; i++)
    printf("Adresse valeur réelle n°%d : %p\tvaleur réelle n°%d : %.2f\n", i+1,
ptr+i,i+1,*(ptr+i));
  printf("\n****** Affichage des minimum et maximum ******\n");
  printf("contenu de ptrMax : %p\tvaleur de la case pointée par ptrMax : %.2f\n", ptrMax,
*ptrMax);
  printf("contenu de ptrMin : %p\tvaleur de la case pointée par ptrMin : %.2f\n", ptrMin,
*ptrMin);
  free(ptr);
  return (0);
}
```

TD n°4

Exercice 1

```
#include <stdio.h>
#define MAX 20
int main(void)
{
   int tab[MAX], end_tab, i;

   do
      {
         printf("NB valeurs (max = %d) ? ", MAX);
         scanf("%d", &end_tab);
      } while (end_tab > MAX || end_tab <= 0);
   for (i = 0; i < end_tab; i++)
      {
         printf("nb n°%d ? ", i+1);
         scanf("%d", &tab[i]);
      }
   printf("\n");
   for (i = 0; i < end_tab; i++)
      printf("nb n°%d = %d\n", i+1, tab[i]);
   return (0);
}
```

Exercice 2

```
#include <stdio.h>
#define MAX 20
int main(void)
{
   int tab[MAX], end_tab, i;

   do
      {
         printf("NB valeurs (max = %d) ? ", MAX);
         scanf("%d", &end_tab);
      } while (end_tab > MAX || end_tab <= 0);
   for (i = 0; i < end_tab; i++)
      {
         printf("nb n°%d ? ", i+1);
         scanf("%d", &tab[i]);
      }
   printf("\n");
   for (i = 0; i < end_tab; i++)
      printf("nb n°%d = %d\n", i+1, tab[i]);
   return (0);
}
```

TP n°4

Exercice 1

```
#include <stdio.h>

#define MIN 0
#define MAX 50

int main(void)
{
 int t[MAX], t1[MAX], i, j, nb, effectif;

 //boucle tant que nb n'est pas correct
 do
  {
   printf("Rentrez le nombre d'éléments à saisir (nb <= %d): ", MAX);
   scanf("%d", &nb);
  } while (nb > MAX || nb <= MIN);
 printf("****Les chiffres doivent être rentrés par ordre croissant ****\n");
 //on entre les nombres
 for(i = 0; i < nb; i++)
  {
   //on vérifie a chaque nb entré qu'il est >= au nb d'avant
   do
    {
     printf("Entrez le chiffre n° %d : ", i+1);
     scanf("%d", &t[i]);
    } while (i > 0 && (t[i-1] > t[i]));
  }
 printf("\n");
 printf("******Affichage du tableau initial *******\nvecteur t:");
 for(i = 0; i < nb; i++)
  {
   printf("\t%d", t[i]);
  }
 i = 0;
 printf("\n");
 printf("\n");
 printf("******Affichage du tableau après et des effectifs ******\nvecteur t1:");
 for(j = 0; i < nb; j++)
  {
   t1[j] = t[i]; //on entre automatiquement la 1ère valeur
   do
    i++; //on incrémente forcément le i car on vient d'utiliser cette case du tableau
   while (i < MAX && (t[i-1] == t[i]));
   printf("\t%d", t1[j]);
  }
 printf("\neffectif  :");
 i = 0;
 effectif = 0;
```

```
  for(j = 0; i < nb; j++)
   {
     do
      {
        effectif++;
        i++;
      }
     while (i < MAX && (t[i] == t1[j]));
     printf("\t%d", effectif);
     effectif = 0; //on remet effectif à 0 pour la prochaine valeur redondante
   }
  printf("\n");
  return (0);
}
```

Exercice 2

```
#include <stdio.h>

#define MAX 30

int main(void)
{
  int t[MAX], tmp, i, j, nb;
  printf("Rentrez le nombre d'éléments à saisir (nb <= %d): ", MAX);
  scanf("%d", &nb);
  for (i = 0; i < nb; i++)
   {
     printf("Entrer t[%d] : ", i);
     scanf("%d", &t[i]);
   }
  printf("\n\n");
  printf("******\tAffichage du tableau initial\t******\n\nvecteur t:\t");
  for(i = 0; i < nb; i++)
    printf("\t%d", t[i]);
  for(i = 0, j = nb-1; i < j; i++, j--)
   {
     tmp = t[i];
     t[i] = t[j];
     t[j] = tmp;
   }
  printf("\n\n******\tAffichage du tableau après inversion\t******\n\nvecteur t inversé:");
  for(i = 0; i < nb; i++)
    printf("\t%d", t[i]);
  printf("\n");
}
```

Exercice 3

```
#include <stdio.h>

#define MAXL 10
#define MAXC 10

int main(void)
{
  int matrice[MAXL][MAXC], nbl, nbc, i, j;
  do
    {
      printf("Nb lignes dans la matrice (<=%d) : ", MAXL);
      scanf("%d", &nbl);
    } while (nbl <= 0 || nbl > MAXL);
  do
    {
      printf("Nb colonnes dans la matrice (<=%d) : ", MAXC);
      scanf("%d", &nbc);
    } while (nbc<= 0 || nbc > MAXC);

  for(i = 0; i < nbl; i++)
    {
      for(j = 0; j < nbc; j++)
        {
          printf("Valeur de matrice[%d][%d] : ", i,j);
          scanf("%d", &matrice[i][j]);
        }
    }
  printf("\n******\tAffichage de la matrice\t******\n");
  for(i = 0; i < nbl; i++)
    {
      for(j = 0; j < nbc; j++)
        printf("\t%d", matrice[i][j]);
      printf("\n");
    }
  printf("\n******\tAffichage de la matrice transposée\t******\n");
  //pour afficher la matrice transposée, on inverse simplement les lignes et les colonnes
  for(j = 0; j < nbc; j++)
    {
      for(i = 0; i < nbl; i++)
        printf("\t%d", matrice[i][j]);
      printf("\n");
    }
}
```

TD n°5

Exercice 1

a)

```
void fonctionB (int x, float f)
{
   /*
    *
    * */
}
int fonctionA (int x, int y)
{
   int z;
   float a;
   /*
    *
    * */
   fonctionB(x,a);
}

int main(void)
{
   int x, y, z;

   z = fonctionA(x,y);
   return (0);
}
```

b)

```
fonctionA(int x, int y);
fonctionB(int x, float f);

int main(void)
{
   int x, y, z;

   z = fonctionA(x,y);
   return (0);
}
void fonctionB (int x, float f)
{
   /*
    *
    * */
}
int fonctionA (int x, int y)
{
   int z;
   float a;
   /*
    *
    * */
   fonctionB(x,a);
}
```

Exercice 2

1)

```
int somme(int x, int y)
{
        return (x + y);
}
```

2)

```
void somme(int *x, int *y, int *result)
{
        *result = *x + *y;
}
```

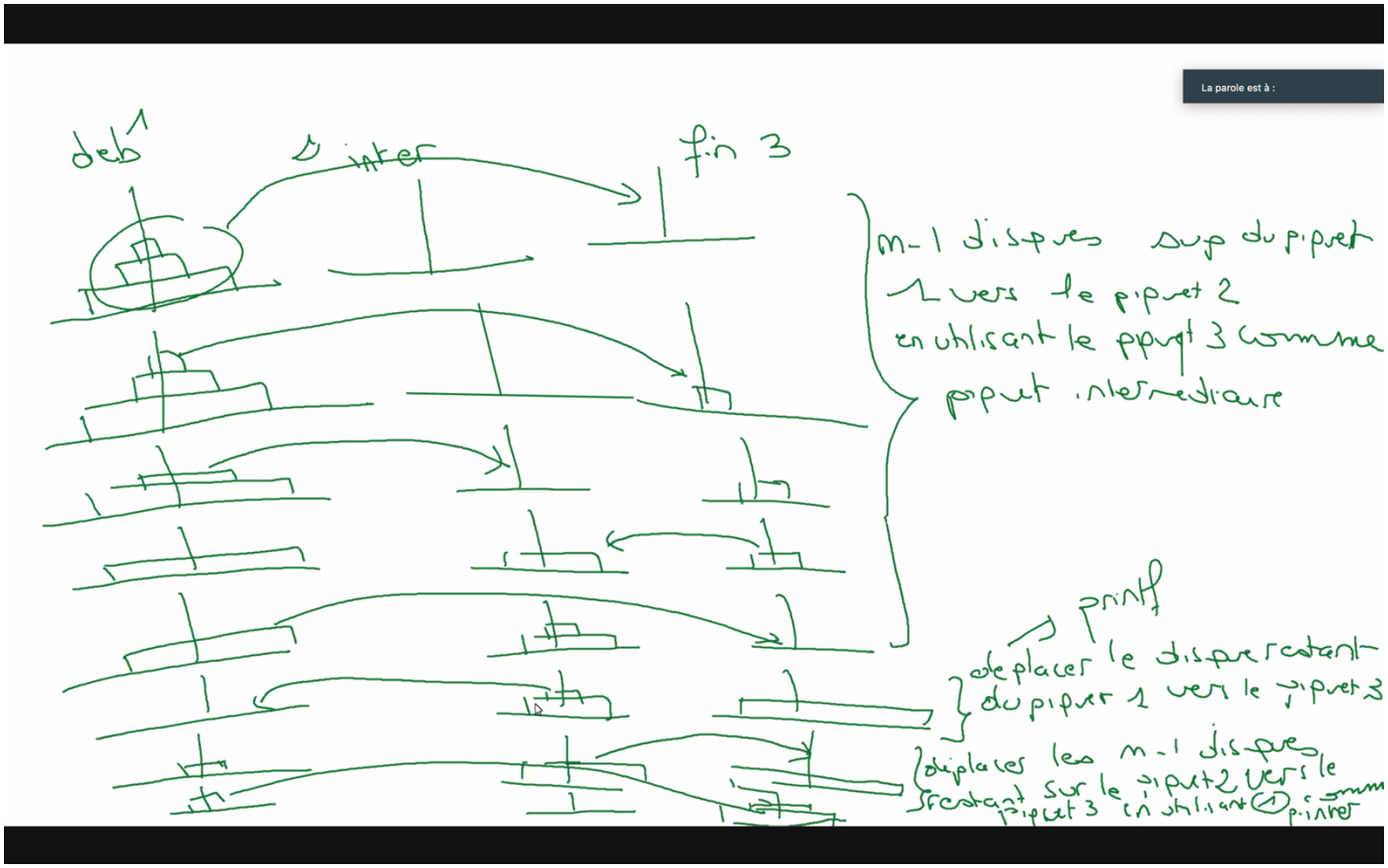
Exercice 3

```
int factorielle_itérative(int n)
{
   int i, result = 1;

   for(i = 0; i < n; i++)
      result *= (n - i);
   return (result);
}
```

```
int factorielle_récursive(int n)
{
   if (n > 0)
      return (n * factorielle_récursive(n - 1));
   else
      return (1);
}
```

Exercice 4

Explications (😭):



```
#include <stdio.h>
#include <stdlib.h>

void hanoi(int nb, int deb, int fin, int inter)
{
   if (nb > 0)
      {
         hanoi(nb -1, deb, inter, fin);
         printf("deplacer disque de %d vers %d\n", deb, fin);
         hanoi(nb -1, inter, fin, deb);
      }
}

int main(void)
{
   int nb;

   printf("nb ? ");
   scanf("%d", &nb);
   hanoi(nb, 1, 3, 2);
}
```

TP n°5

Exercice 1

```
#include <stdio.h>

int mult_2(int nb)
{
   if ((nb%2) == 0)
      return (1);
   else
      return (0);
}

int mult_3(int nb)
{
   if ((nb%3) == 0)
      return (1);
   else
      return (0);
}

int main(void)
{
   int nb;

   printf("Nb entier ? ");
   scanf("%d", &nb);
   printf("%d est ", nb);
   if (mult_2(nb) == 1)
      printf("pair");
   else if (mult_3(nb) == 1)
      printf("un multiple de 3");
   if ((nb % 6) == 0)
      printf(" et un multiple de 6");
   printf("\n");
   return (0);
}
```

Exercice 2

```
#include <stdio.h>

void    swap(int *x, int *y)
{
   int tmp;

   tmp = *x;
   *x = *y;
   *y = tmp;
   printf("before end of swap : x = %d, y = %d\n", *x, *y);
}

int     main(void)
{
   int param1, param2;

   printf("Parametres ? ");
   scanf("%d%d", &param1, &param2);
   printf("Before swap : x = %d; y = %d\n", param1, param2);
   swap(&param1, &param2);
   printf("After swap : x = %d; y = %d\n", param1, param2);
   return (0);
}
```

Exercice 3

```
#include <stdio.h>

void min_max(int x, int y, int z, int *min, int *max)
{
   *min = x;
   *max = x;
   if (*min > y)
      *min = y;
   if (*min > z)
      *min = z;
   if (*max < y)
      *max = y;
   if (*max < z)
      *max = z;
}

int main(void)
{
   int x,y,z,min, max;

   printf("valeur 1 : ");
   scanf("%d", &x);
```

```
   printf("valeur 2 : ");
   scanf("%d", &y);
   printf("valeur 3 : ");
   scanf("%d", &z);
   min_max(x,y,z,&min,&max);
   printf("min = %d, max = %d\n", min, max);
   return (0);
}
```

Exercice 4

```
#include <stdio.h>

int coeffBinome(int n, int k)
{
   if (n >= 2 && k > 0 && k < n)
      {
         return (coeffBinome(n-1,k-1) + coeffBinome(n-1,k));
      }
   else
      return (1);

}

int factorielle(int n)
{
   if (n > 1)
      return (factorielle(n - 1) * n);
   else
      return (1);
}

int coeffBinome_iterative(int n, int p)
{
   return (factorielle(n) / (factorielle(p) * (factorielle(n-p))));
}

int puissance(int x, int y)
{
   int i,result=1;

   for(i = y; i > 0; i--)
      result *= x;
   return (result);
}

int binome_newton(int x, int y, int n)
{
   int p;
   int result=0;
```

```
   for(p = 0; p <= n; p++)
      result += coeffBinome(n,p) * puissance(x,n-p) * puissance(y,p);
   return (result);
}

int main(void)
{
   int x,y,n;

   printf("x ? ");
   scanf("%d", &x);
   printf("y ? ");
   scanf("%d", &y);
   printf("n ? ");
   scanf("%d", &n);
   printf("(%d + %d)^%d = %d\n", x,y,n,binome_newton(x,y,n));
   return (0);
}
```

TD n°6

Exercice 1