

---

# Titre du TIPE sur Jeux et Sports

## *Code du TIPE*

---

Ceci est un modèle pour générer un PDF qui présente le code d'un TIPE.

Le fichier principal `rendu_code.tex` est à compiler avec l'option `-shell-escape`.

Conseils :

- Commencez par faire du ménage dans votre code, supprimer les fichiers obsolètes, les brouillons... Si ce n'est pas fait, c'est peut-être aussi l'occasion de séparer son code en plusieurs fichiers...
- Placez le dossier avec ce code latex à proximité de votre code dossier de code (pas dedans) et indiquez les fichiers de code à inclure grâce à un chemin relatif. Typiquement si votre dossier TIPE contient les dossiers `Code` et `rendu_code`, vous écrirez `../code/sous_dossier/fichier.c`. Il est déconseillé de copier tout son code dans un dossier spécial pour générer le rendu de code, cela n'est pas très robuste aux modifications et ajouts de code ultérieurs.
- Si vous importez le code du fichier en entier, pensez à supprimer les lignes vides en fin de fichier, les portions de code commenté devenu inutile...
- Si au contraire vous utilisez l'import par blocs, il est conseillé de sauter des lignes entre les différentes fonctions de sorte que chaque bloc commence à une ligne multiple de 10 (par exemple). Cela permet, dans le cas où vous modifiez a posteriori une fonction, en ajoutant une ligne de commentaire par exemple, d'avoir à modifier seulement les numéros de lignes du bloc correspondant, et pas ceux de tous les blocs suivants.
- Pensez à ajuster le modèle pour une version finale :
  - renseignez nom, prénom et numéro de candidat dans le fichier `mise_en_forme_MPI.tex`, pour la première page l.6 ET pour le pied de page l.32 ;
  - adaptez le titre et la date ;
  - supprimez cette section de conseils en commentant la l.20 ;
  - à la fin il ne doit plus y avoir de rouge dans le PDF.
- Compilez plusieurs fois si nécessaire pour ajuster le nombre total de pages en pied de page.
- Si vous ne réduisez pas la police, vous pouvez imprimer ce PDF livret (A4-> A5, en 2 pages par feuille...), ça reste lisible.

# 1 Code sur la première partie

## 1.1 Code du fichier nom\_fichier.c

ici, on insère le fichier entier, tel quel, pas forcément adapté.

```
1  #include <stdio.h>
2  #include <stdbool.h>
3  #include <assert.h>
4
5  void type_simple_triangle (int a, int b, int c){
6  //hyp : a>=0 && b>=0 && c>=0
7      if (a==b){
8          if(b==c)
9              printf("équilatéral");
10         else
11             printf("isocèle");
12     }
13     else{
14         if((b==c) || (a==c))
15             printf("isocèle");
16         else
17             printf("scalène");
18     }
19     printf("\n");
20 }
21
22
23
24
25 bool est_triangle(int a, int b, int c){
26     return (a<=b+c) && (b<=a+c) && (c<=a+b) ;
27 }
28
29
30
31
32
33
34
35 bool est_plat(int a, int b, int c){
36     return (a==b+c) || (b==a+c) || (c==a+b) ;
37 }
38
39
40
41
42
43
44
45 bool est_rectangle(int a, int b, int c){
46     return (a*a + b*b == c*c) || (b*b + c*c == a*a) || (a*a + c*c == b*b) ;
47 }
```

```

48
49
50
51
52
53
54
55 void affiche_type_triangle(int a,int b, int c){
56     if (!est_triangle(a,b,c))
57         printf("Ces longueurs ne sont les 3 côtés d'un même triangle\n");
58     else{
59         if (a==b){
60             if(b==c){
61                 if (a==0) // alors a=b=c=0
62                     printf("équilatéral plat = réduit à un point");
63                 else
64                     printf("équilatéral");
65             }
66             else if(2*b*b == c*c) // dans ce cas  $a^2+b^2=c^2$ 
67                 printf("isocèle rectangle");
68             else if (a+b == c)
69                 printf("isocèle plat");
70             else
71                 printf("isocèle");
72         }
73         else{ // ici on sait que a!=b
74             if (b == c) {
75                 if(2*b*b == a*a) // dans ce cas  $c^2+b^2=a^2$ 
76                     printf("isocèle rectangle");
77                 else if (b+c == a)
78                     printf("isocèle plat");
79                 else
80                     printf("isocèle");
81             }
82             else if (a == c){
83                 // ici on sait que a!=b et b!=c et a=c
84                 if(2*a*a == b*b) // dans ce cas  $a^2+c^2=b^2$ 
85                     printf("isocèle rectangle");
86                 else if (a+c == b)
87                     printf("isocèle plat");
88                 else
89                     printf("isocèle");
90             }
91             else{ // ici on sait que a!=b et b!=c et a!=c
92                 if (est_plat(a,b,c))
93                     printf("plat");
94                 else if (est_rectangle(a,b,c))
95                     printf("rectangle");
96                 else
97                     printf("scalène");
98             }
99         }

```

```

100     printf("\n");
101 }
102 }
103
104
105 int main(){
106     // jeu de test de type_triangle
107     type_simple_triangle (1,1,1); //equi
108     type_simple_triangle (1,1,2); //iso
109     type_simple_triangle (1,2,1); //iso
110     type_simple_triangle (2,1,1); //iso
111     type_simple_triangle (1,2,3); //scalene
112
113
114
115     // jeu de test de est_triangle
116     assert(est_triangle (1,2,3));
117     assert(est_triangle (3,4,5));
118     assert(est_triangle (1,1,1));
119     assert(!est_triangle (3,4,15));
120     assert(!est_triangle (31,4,15));
121
122
123
124
125     // jeu de test de est_plat
126     assert(est_plat (1,2,3));
127     assert(!est_plat (3,4,5));
128
129
130     // jeu de test de est_rectangle
131     assert(!est_rectangle (1,2,3));
132     assert(est_rectangle (3,4,5));
133
134
135     // jeu de test de affiche_type_triangle
136     affiche_type_triangle (1,1,1); //equi
137     affiche_type_triangle (0,0,0); // equi plat = pt
138     affiche_type_triangle (1,1,2); //iso plat
139     affiche_type_triangle (1,2,1); //iso plat
140     affiche_type_triangle (2,1,1); //iso plat
141     affiche_type_triangle (3,3,5); //iso
142     affiche_type_triangle (1,2,3); //plat
143     affiche_type_triangle (3,4,5); //rectangle
144     affiche_type_triangle (3,4,6); //scalène
145     //isocèle rectangle n'arrive pas avec des longueurs entières
146
147     return 0;
148 }

```

## 1.2 Code du fichier nom\_fichier.c

ici, on insère le fichier par bloc, de telle ligne à telle ligne. On utilise même `\newpage` pour éviter que le bloc dumain soit à cheval sur deux pages

```
1  #include <stdio.h>
2  #include <stdbool.h>
3  #include <assert.h>
4
5  void type_simple_triangle (int a, int b, int c){
6  //hyp : a>=0 && b>=0 && c>=0
7      if (a==b){
8          if(b==c)
9              printf("équilatéral");
10         else
11             printf("isocèle");
12     }
13     else{
14         if((b==c) || (a==c))
15             printf("isocèle");
16         else
17             printf("scalène");
18     }
19     printf("\n");
20 }
21
22
23
24
25 bool est_triangle(int a, int b, int c){
26     return (a<=b+c) && (b<=a+c) && (c<=a+b) ;
27 }
28
29
30
31
32
33
34
35 bool est_plat(int a, int b, int c){
36     return (a==b+c) || (b==a+c) || (c==a+b) ;
37 }
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55 void affiche_type_triangle(int a,int b, int c){
56     if (!est_triangle(a,b,c))
57         printf("Ces longueurs ne sont les 3 côtés d'un même triangle\n");
58     else{
59         if (a==b){
60             if(b==c){
61                 if (a==0) // alors a=b=c=0
62                     printf("équilatéral plat = réduit à un point");
63                 else
64                     printf("équilatéral");
65             }
66             else if(2*b*b == c*c) // dans ce cas a²+b²=c²
67                 printf("isocèle rectangle");
68             else if (a+b == c)
69                 printf("isocèle plat");
70             else
71                 printf("isocèle");
72         }
73         else{ // ici on sait que a!=b
74             if (b == c) {
```

```

75     if(2*b*b == a*a) // dans ce cas  $c^2+b^2=a^2$ 
76         printf("isocèle rectangle");
77     else if (b+c == a)
78         printf("isocèle plat");
79     else
80         printf("isocèle");
81 }
82 else if (a == c){
83     // ici on sait que  $a \neq b$  et  $b \neq c$  et  $a=c$ 
84     if(2*a*a == b*b) // dans ce cas  $a^2+c^2=b^2$ 
85         printf("isocèle rectangle");
86     else if (a+c == b)
87         printf("isocèle plat");
88     else
89         printf("isocèle");
90 }
91 else{ // ici on sait que  $a \neq b$  et  $b \neq c$  et  $a \neq c$ 
92     if (est_plat(a,b,c))
93         printf("plat");
94     else if (est_rectangle(a,b,c))
95         printf("rectangle");
96     else
97         printf("scalène");
98 }
99 }
100 printf("\n");
101 }
102 }

```

```

105 int main(){
106     // jeu de test de type_triangle
107     type_simple_triangle (1,1,1); //equi
108     type_simple_triangle (1,1,2); //iso
109     type_simple_triangle (1,2,1); //iso
110     type_simple_triangle (2,1,1); //iso
111     type_simple_triangle (1,2,3); //scalene
112
113
114
115     // jeu de test de est_triangle
116     assert(est_triangle (1,2,3));
117     assert(est_triangle (3,4,5));
118     assert(est_triangle (1,1,1));
119     assert(!est_triangle (3,4,15));
120     assert(!est_triangle (31,4,15));
121
122
123
124
125     // jeu de test de est_plat
126     assert(est_plat (1,2,3));
127     assert(!est_plat (3,4,5));
128
129
130     // jeu de test de est_rectangle
131     assert(!est_rectangle (1,2,3));
132     assert(est_rectangle (3,4,5));
133
134
135     // jeu de test de affiche_type_triangle
136     affiche_type_triangle (1,1,1); //equi
137     affiche_type_triangle (0,0,0); // equi plat = pt
138     affiche_type_triangle (1,1,2); //iso plat
139     affiche_type_triangle (1,2,1); //iso plat
140     affiche_type_triangle (2,1,1); //iso plat
141     affiche_type_triangle (3,3,5); //iso
142     affiche_type_triangle (1,2,3); //plat
143     affiche_type_triangle (3,4,5); //rectangle
144     affiche_type_triangle (3,4,6); //scalène
145     //isocèle rectangle n'arrive pas avec des longueurs entières
146
147     return 0;
148 }

```

## 2 Code de la deuxième partie

### 2.1 Code du fichier nom\_fichier.ml

```
1 let oppose (l:int list) : int list =
2   let rec aux (ll:int list) (lres:int list) : int list =
3     match ll with
4     | [] -> lres
5     | a::q -> aux q ((-a)::lres)
6   in List.rev (aux l [])
7
8 let test_oppose : unit =
9   assert(oppose [] = []);
10  assert(oppose [-2; 3; 4; -5] = [2; -3; -4; 5]);
11  assert(oppose [10; -10; 10] = [-10; 10; -10])
12
13
14 let intervertit (l: ('a*'b) list) : ('b*'a) list =
15   let rec aux (ll:('a*'b) list) (lres:('b*'a) list) : ('b*'a) list =
16     match ll with
17     | [] -> lres
18     | (x,y)::q -> aux q ((y,x)::lres)
19   in List.rev (aux l [])
20
21 let test_intervertit : unit =
22   assert (intervertit [] = []);
23   assert (intervertit [(1,2); (3,4)] = [(2,1); (4,3)]);
24   assert (intervertit [('a',1); ('b',3); ('z',18)] = [(1,'a'); (3,'b');
25     ↪ (18,'z')] )
26
27 let demultiplie (l: ('a*int) list) : 'a list =
28   let rec aux (ll:('a*int) list) (lres:'a list) : 'a list =
29     match ll with
30     | [] -> lres
31     | (elem,k)::q -> if k<=0 then aux q lres else aux ((elem,k-1)::q)
32     ↪ (elem::lres)
33   in List.rev (aux l [])
34
35 let test_demultiplie : unit =
36   assert (demultiplie [] = []);
37   assert (demultiplie [('a',0)] = []);
38   assert (demultiplie [(true,3)] = [true; true; true]);
39   assert (demultiplie [('a',1); ('c',3); ('a',2)] = ['a'; 'c'; 'c'; 'c'; 'a';
40     ↪ 'a'])
41
42 let compresse (l:'a list) : ('a*int) list =
43   (*hyp : l <> [] *)
44   let rec aux (ll:'a list) (lres:('a*int) list) (elem: 'a) (nb:int) : ('a*int)
45     ↪ list =
```



```

45     match ll with
46     | [] -> (elem,nb)::lres
47     | a::q -> if a=elem then aux q lres elem (nb+1) else aux q
        ↪ ((elem,nb)::lres) a 1
48 in match l with
49 | [] -> failwith "on ne compresse pas la liste vide"
50 | e1::l1 -> List.rev (aux l1 [] e1 1)
51
52 let test_compresse : unit =
53     assert (compresse [true; true; true] = [(true,3)] );
54     assert (compresse [1; 1; 2; 2; 2; 3; 3] = [(1,2); (2,3); (3,2)] );
55     assert (compresse ['a'; 'c'; 'c'; 'c'; 'a'; 'a'] = [('a',1); ('c',3);
        ↪ ('a',2)] );
56
57
58
59 let moyenne (l:int list) : float =
60     (*hyp : l <> [] *)
61     let rec aux (ll:int list) (somme:int) (cpt:int) : int*int =
62         match ll with
63         | [] -> (somme, cpt)
64         | a::q -> aux q (somme+a) (cpt+1)
65     in let s,c = aux l 0 0
66     in float_of_int(s)/.float_of_int(c)
67
68
69 let test_moyenne : unit =
70     assert (moyenne[1; 1; 1; 1; 1] = 1.);
71     assert (moyenne[1; 2; 3] = 2.);
72     assert (moyenne[-10; 10; 5] = (5./3.))
73
74 let moyennes_listes (l: int list list) : float list =
75     (*hyp : les éléments de l sont non vides *)
76     let rec aux (ll: int list list) (lres:float list) : float list =
77         match ll with
78         | [] -> lres
79         | a::q -> aux q ((moyenne a)::lres)
80     in List.rev (aux l [])
81
82 let test_moyennes_listes : unit =
83     assert (moyennes_listes[] = []);
84     assert (moyennes_listes [[1; 1; 1; 1; 1]; [2;3]] = [1.; 2.5]);
85     assert (moyennes_listes [[1; 2; 3]; [3; 2; 1]; [2; 1; 3]] = [2.; 2.; 2.]);
86     assert (moyennes_listes [[-10; 10; 5]] = [5./3.])
87
88
89 let moyenne_bis (l:int list) : float option =
90     match l with
91     | [] -> None
92     | _ ->
93         let rec aux (ll:int list) (somme:int) (cpt:int) : int*int =
94             match ll with

```

```

95     | [] -> (somme, cpt)
96     | a::q -> aux q (somme+a) (cpt+1)
97 in let s,c = aux l 0 0
98 in Some (float_of_int(s)/.float_of_int(c))
99
100
101 let moyennes_listes_bis (l: int list list) : float list =
102   (*hyp : les éléments de l sont non vides *)
103   let rec aux (ll: int list list) (lres:float list) : float list =
104     match ll with
105     | [] -> lres
106     | a::q -> let m = (moyenne_bis a) in
107       match m with
108       | None -> aux q lres
109       | Some moy -> aux q (moy::lres)
110 in List.rev (aux l [])
111
112
113 let test_moyennes_listes_bis : unit =
114   assert (moyennes_listes_bis [] = []);
115   assert (moyennes_listes_bis [[1; 1; 1; 1; 1]; []; [2;3]] = [1.; 2.5]);
116   assert (moyennes_listes_bis [[1; 2; 3]; [3; 2; 1]; [2; 1; 3]; [];[]] = [2.;
117     ↪ 2.; 2.]);
118   assert (moyennes_listes_bis [[-10; 10; 5]; [0]; []] = [5./3.; 0.])
119
120 let moyennes_listes_ter (l: int list list) : float list =
121   List.filter_map moyenne_bis l
122
123 let test_moyennes_listes_ter : unit =
124   assert (moyennes_listes_ter [] = []);
125   assert (moyennes_listes_ter [[1; 1; 1; 1; 1]; []; [2;3]] = [1.; 2.5]);
126   assert (moyennes_listes_ter [[1; 2; 3]; [3; 2; 1]; [2; 1; 3]; [];[]] = [2.;
127     ↪ 2.; 2.]);
128   assert (moyennes_listes_ter [[-10; 10; 5]; [0]; []] = [5./3.; 0.])

```