Exercice 1. 1. Exprimer -53_{10} en CA2 sur 8 bits selon la méthode du cours [1] ;

2. Exprimer $\overline{1100\,1101}^{8}$ en base 10.

Démonstration. Voici $-53 \to \overline{11001011}^{8}$
dans l'autre sens -51.

□

Exercice 2. Ecrire la fonction `alterne (n:int) : int` qui calcule la somme

$$\sum_{i=0}^{n} i^{1+\varepsilon(i)}$$

où $\varepsilon(i) = 0$ si i est paire et 1 sinon.

Démonstration. Code

```
let rec alterne n =
  if n = 0 then 0
  else
    let i = n in
    if i mod 2 = 0 then
      i + alterne (n - 1)
    else
      (i * i) + alterne (n - 1)
```

□

Exercice 3. Écrire une fonction `is_sorted (l:'a list) : bool` qui renvoie vrai si et seulement si la liste `l` est triée par ordre croissant ou décroissant.

Démonstration.
```
let is_sorted l =
  let rec aux b acc l = match l with
      [] | [_] -> b, acc
      | x::y::q -> if x > y
                   then aux false (x::acc) q
                   else aux b (x::acc) (y::q)
  in let rec aux2 l = match l with
        [] | [_] -> true
        | x::y::q -> if x > y
                     then false
                     else aux2 (y::q)
     in let b,acc = aux true [] l in
        b || aux2 acc;;
```

□

Exercice 4. Écrire la fonction OCaml `triangle_gauche (n:int) : unit` qui affiche un triangle isocèle à n lignes avec un côté vertical à gauche.

1. On peut utiliser une autre méthode mais il faut montrer pourquoi elle est correcte (et de toute façon, ça ne dispense pas de connaître son cours).

```
# triangle_gauche 5;;
*
**
***
****
*****
- : unit = ()
```

Démonstration. Code

```
let triangle_gauche n =
  let rec aux numero =
    if numero=n then ()
    else (
      let rec print_line m  =
        (*m nb total, k nb souhaité*)
        if m = numero+1 then ()
        else (
          print_string "*";
          print_line (m+1)
        ) in
      print_line 0;
      print_string "\n";
      aux (numero+1)
    )
  in aux 0;;
```

□

Exercice 5. Écrire la fonction `add_vect (v1: int list) (v2 : int list)` qui réalise l'addition de deux vecteurs à coordonnées entières. Si les dimensions ne sont pas les mêmes, une exception est soulevée.

Aucune fonction du module liste n'est autorisée.

Démonstration. Code

```
exception Longueurs_differentes;;

(* 1) Somme de deux vecteurs : add_vect v1 v2 *)

let rec add_vect v1 v2 =
  match v1, v2 with
  | [], [] ->
      []
  | x1 :: q1, x2 :: q2 ->
      (x1 + x2) :: add_vect q1 q2
  | _ ->
      raise Longueurs_differentes
;;
```

□

Exercice 6.

Rappel. Soit (A, B, C) un triangle strict du plan. Un point P peut s'écrire comme combinaison des trois sommets :

$$P = \omega_1 A + \omega_2 B + \omega_3 C$$

avec la contrainte $\omega_1 + \omega_2 + \omega_3 = 1$.

Si tous les coefficients sont positifs ou nuls, alors P est dans le triangle ou sur les bords. Si un des coefficients est négatif, alors p est hors du triangle.

On note d le double de l'aire algébrique du triangle :

$$d = (B_x - C_x)(A_y - C_y) + (C_y - B_y)(A_x - C_x)$$

On a

$$\begin{aligned} \omega_1 &= \frac{(B_x - C_x)(P_y - C_y) + (C_y - B_y)(P_x - C_x)}{d} \\ \omega_2 &= \frac{(C_x - A_x)(P_y - C_y) + (A_y - C_y)(P_x - C_x)}{d} \\ \omega_3 &= 1 - \omega_1 - \omega_2 \end{aligned}$$

Format des fichiers PBM. Un fichier `PBM` (*Portable BitMap*) est un format texte très simple utilisé pour représenter des images en noir et blanc. Le fichier commence par une ligne contenant le mot magique `P1`, qui indique qu'il s'agit d'une image PBM au format ASCII. La deuxième ligne contient deux entiers : la **largeur** (nombre de colonnes) puis la **hauteur** (nombre de lignes). Viennent ensuite les valeurs des pixels, rangées ligne par ligne, séparées par des espaces ou des retours à la ligne.

Chaque pixel est codé par un seul chiffre :

$$1 \text{ pour le noir}, \qquad 0 \text{ pour le blanc}.$$

Par exemple, une image de 3×2 pixels entièrement noire est représentée par :

```
P1
3 2
1 1 1
1 1 1
```

On veut lire et écrire dans des fichiers au format PBM. On donne le fichier d'en-tête **pbm.h** :

```c
#ifndef PBM_H
#define PBM_H
typedef struct { int i, j; } point;
typedef struct {
    int h, w;
    unsigned char *data;
} pbm;

pbm *load(const char *filename);
pbm *neg(const pbm *img);
void save(const pbm *img, const char *filename);
pbm *triangle(int w, int h, point p1, point p2, point p3);//noir sur fond blanc
pbm *rotate(const pbm *img, point A, float theta);
void free_pbm(pbm *img);

static inline unsigned char getpx(const pbm *img, int i, int j) {
    return img->data[i*img->w + j];
}
static inline void setpx(pbm *img, int i, int j, unsigned char v) {
    img->data[i*img->w + j] = v ? 1 : 0;
}
#endif
```

Enfin, on donne un `main` :

```c
#include <stdio.h>
#include <math.h>
#ifndef M_PI
#define M_PI 3.14159265358979323846
#endif
#include "pbm.h"

int main(void) {
    point a = {5, 10}, b = {25, 5}, c = {20, 30};
    pbm *t = triangle(40, 30, a, b, c);
    save(t, "triangle.pbm");

    pbm *tn = neg(t);
    save(tn, "triangle_neg.pbm");

    pbm *tr = rotate(t, a, (float)M_PI/6.0f); // ou acosf(-1.0f)/6.0f
    save(tr, "triangle_rot.pbm");

    free_pbm(tr);
    free_pbm(tn);
    free_pbm(t);
    return 0;
}
```

et un **Makefile**

```
all : pbm_demo

pbm_demo : main.c pbm.c pbm.h
	$(CC) -O2 -std=c17 -lm -o pbm_demo main.c pbm.c

clean :
	rm -f pbm_demo triangle.pbm triangle_neg.pbm triangle_rot.pbm
```

1. Écrire la fonction `pbm *load(const char *filename)` qui prend en paramètre un nom de fichier PBM et charge son contenu dans un pointeur sur `pbm`.
2. Écrire la fonction `void save(const pbm *img, const char *filename);` qui prend en paramètre un pointeur sur `pbm` et crée une image PBM respectant le contenu de l'objet pointé.
3. Écrire la fonction `pbm *neg(const pbm *img);` qui calcule le négatif d'une image décrite par un pointeur sur `pbm`.
4. Écrire la fonction `pbm *triangle(int w, int h, point p1, point p2, point p3);` qui crée un triangle noir sur fond blanc. Les sommets du triangle sont les trois points passés en paramètre.
5. Écrire la fonction `pbm *rotate(const pbm *img, point A, float theta);` qui prend en paramètre un pointeur sur `pbm` décrivant une image. La fonction applique une rotation de centre A (le point sur lequel pointe `A`) et d'angle θ.

Démonstration. Code

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "pbm.h"

pbm *load(const char *filename) {
    FILE *f = fopen(filename, "r");
    char magic[3]; int w, h;
    fscanf(f, "%2s", magic);   // "P1"
    // On saute les commentaires éventuels
    int c = fgetc(f);
    while (c == '#') { while ((c = fgetc(f)) != '\n'); c = fgetc(f); }
    ungetc(c, f);
    fscanf(f, "%d %d", &w, &h);

    pbm *img = malloc(sizeof *img);
    img->w = w; img->h = h;
    img->data = malloc((size_t)w*h);
    for (int i = 0; i < h; i++)
        for (int j = 0; j < w; j++) {
            int v; fscanf(f, "%d", &v);
            setpx(img, i, j, (unsigned char)v);
        }
    fclose(f);
    return img;
}

void save(const pbm *img, const char *filename) {
    FILE *f = fopen(filename, "w");
    fprintf(f, "P1\n%d %d\n", img->w, img->h);
    for (int i = 0; i < img->h; i++) {
        for (int j = 0; j < img->w; j++) {
            fprintf(f, "%d ", getpx(img, i, j));
        }
        fprintf(f, "\n");
    }
    fclose(f);
}

pbm *neg(const pbm *img) {
    pbm *out = malloc(sizeof *out);
    out->w = img->w; out->h = img->h;
    out->data = malloc((size_t)out->w*out->h);
    for (int i = 0; i < out->h; i++)
        for (int j = 0; j < out->w; j++)
            setpx(out, i, j, 1 - getpx(img, i, j));
    return out;
}

// aide : test point dans triangle par barycentres
static int inside_triangle(point p, point a, point b, point c) {
    float denom = (float)((b.j - c.j)*(a.i - c.i) + (c.i - b.i)*(a.j - c.j));
    float w1 = ((b.j - c.j)*(p.i - c.i) + (c.i - b.i)*(p.j - c.j)) / denom;
    float w2 = ((c.j - a.j)*(p.i - c.i) + (a.i - c.i)*(p.j - c.j)) / denom;
    float w3 = 1.0f - w1 - w2;
    return (w1 >= 0 && w2 >= 0 && w3 >= 0);
}

pbm *triangle(int w, int h, point p1, point p2, point p3) {
    pbm *img = malloc(sizeof *img);
    img->w = w; img->h = h;
```

```
63      img->data = malloc((size_t)w*h);
64      // fond noir
65      memset(img->data, 0, (size_t)w*h);
66      // triangle blanc
67      for (int i = 0; i < h; i++)
68          for (int j = 0; j < w; j++) {
69              point p = { i, j };
70              if (inside_triangle(p, p1, p2, p3)) setpx(img, i, j, 1);
71          }
72      return img;
73  }
74
75  pbm *rotate(const pbm *img, point A, float theta) {
76      pbm *out = malloc(sizeof *out);
77      out->w = img->w; out->h = img->h;
78      out->data = malloc((size_t)out->w*out->h);
79      memset(out->data, 0, (size_t)out->w*out->h);
80      float ct = cosf(theta), st = sinf(theta);
81
82      for (int i = 0; i < out->h; i++)
83          for (int j = 0; j < out->w; j++) {
84              // coordonnées d'origine (rotation inverse)
85              float y = i - A.i;
86              float x = j - A.j;
87              float yo =  ct*y + st*x;
88              float xo = -st*y + ct*x;
89              int si = (int)lroundf(yo + A.i);
90              int sj = (int)lroundf(xo + A.j);
91              if (si >= 0 && si < img->h && sj >= 0 && sj < img->w) {
92                  setpx(out, i, j, getpx(img, si, sj));
93              }
94          }
95      return out;
96  }
97
98  void free_pbm(pbm *img) {
99      free(img->data);
100     free(img);
101 }
```

□