Tache 7 Partie 1

File geom2d.h :

```
...
typedef struct Bezier2_
{
   Point A, B, C;
} Bezier2;

typedef struct Bezier3_
{
   Point A, B, C, D;
} Bezier3;
...
```

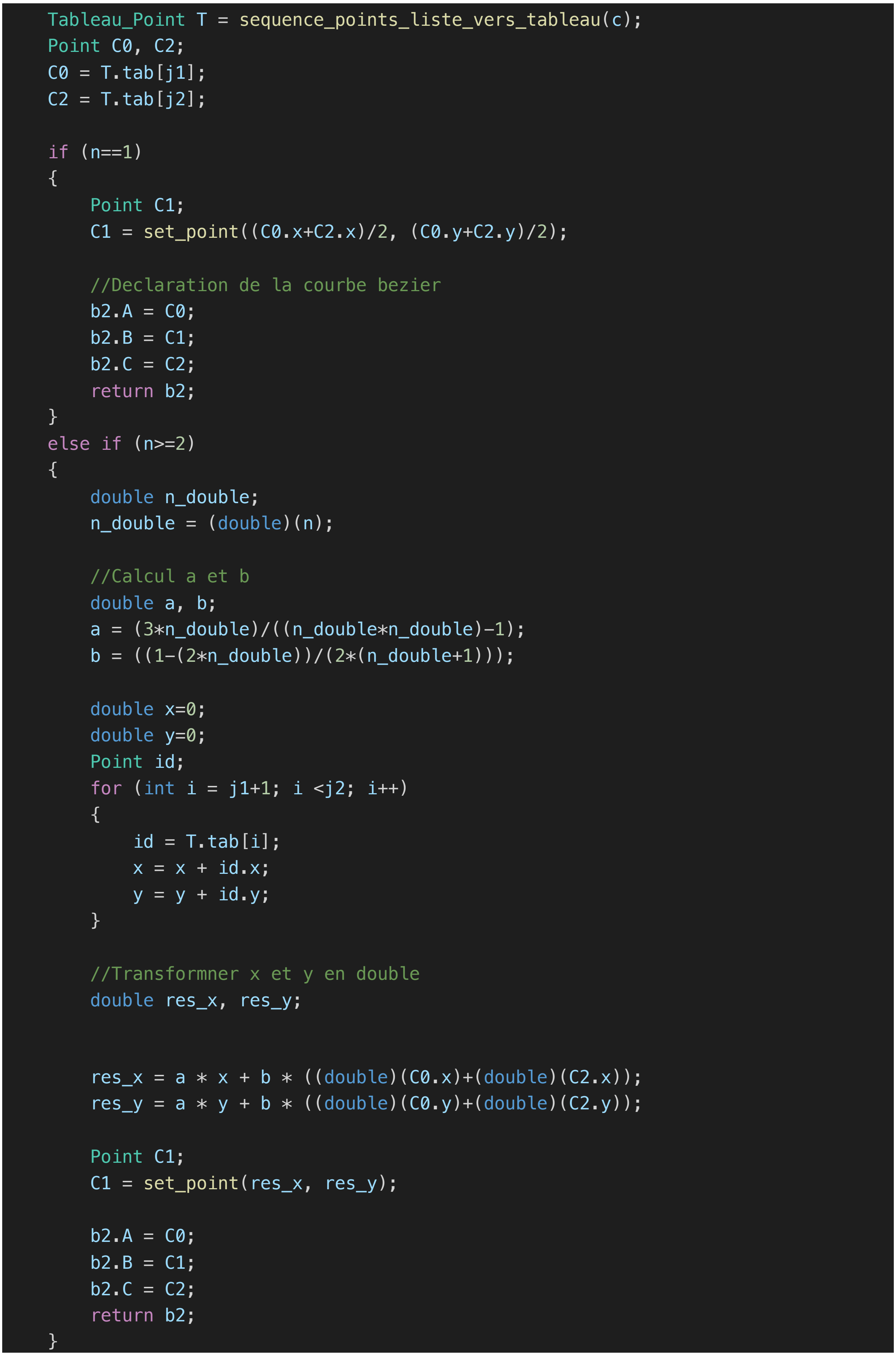
File contour.h:

```
...
Point calcul_ct_bezier2(Bezier2 b2, double t);

Point calcul_ct_bezier3(Bezier3 b3, double t);

Bezier3 conversion_bezier2_to_bezier3 (Bezier2 b2);

Bezier2 approx_bezier2(Contour c, int j1, int j2);

double distance_point_bezier2(Point P1, Bezier2 b2, double ti);

Contour simplification_douglas_peucker_bezier2(Contour C, int j1, int j2,double d);

void create_postscript_contours_bezier2(Liste_Contours c, char *file_name, int
hauteur, int largeur);
...
```

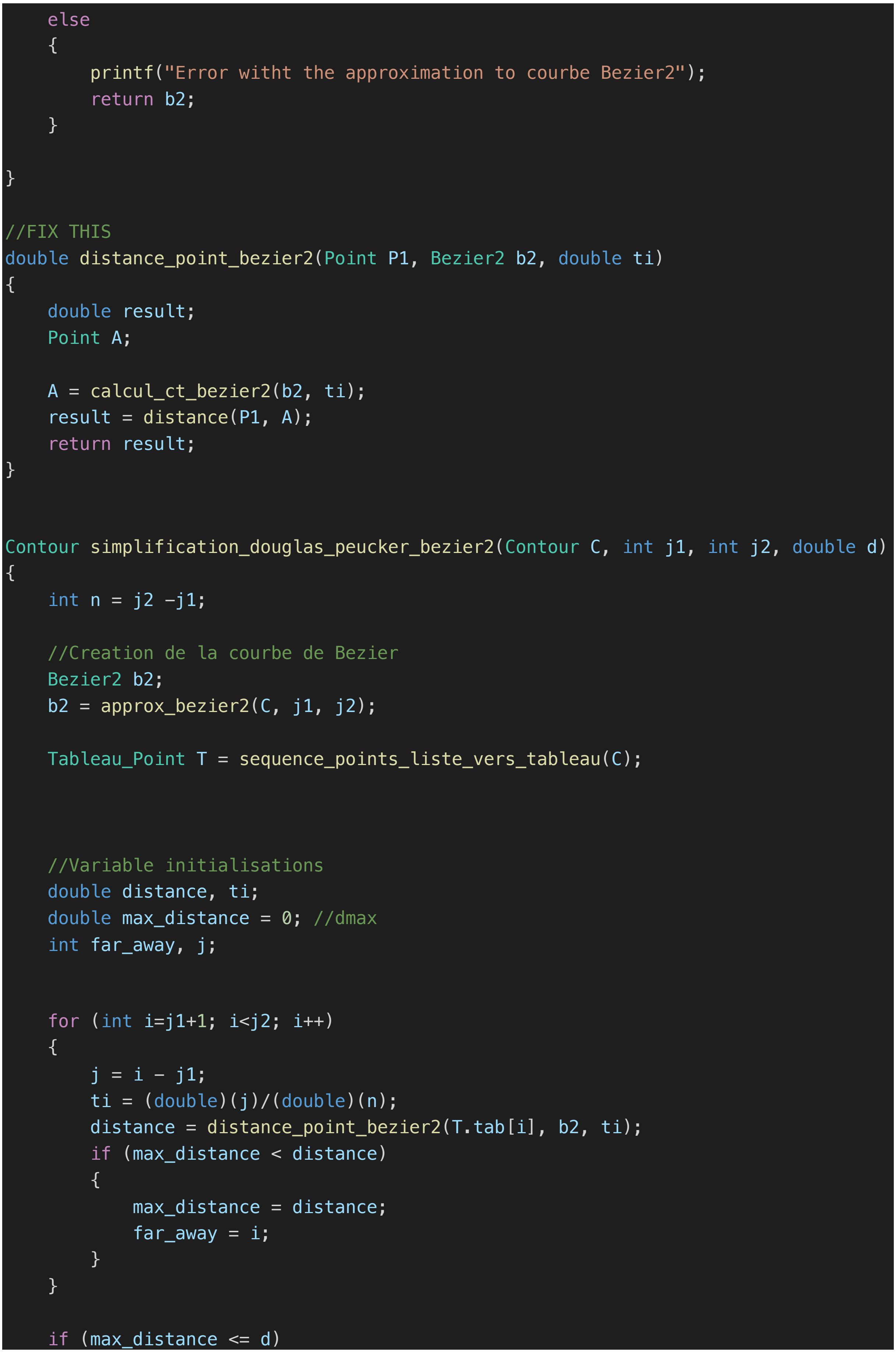
Source code de contour.c modifié :

```
...
Point calcul_ct_bezier2(Bezier2 b2, double t)
{
    Point A;
    int x,y;
    x = ((1-t)*(1-t)*(b2.A.x))+(2*t*(1-t)*(b2.B.x))+(t*t*(b2.C.x));
    y = ((1-t)*(1-t)*(b2.A.y))+(2*t*(1-t)*(b2.B.y))+(t*t*(b2.C.y));
    A = set_point(x,y);
    return A;
}
```

```
Point calcul_ct_bezier3(Bezier3 b3, double t)
{
    Point A;
    double x,y;
    x = ((1-t)*(1-t)*(1-t)*(b3.A.x))+(3*t*(1-t)*(1-t)*(b3.B.x))+(3*t*t*(1-
t)*(b3.C.x))+(t*t*t*(b3.D.x));
    y = ((1-t)*(1-t)*(1-t)*(b3.A.y))+(3*t*(1-t)*(1-t)*(b3.B.y))+(3*t*t*(1-
t)*(b3.C.y))+(t*t*t*(b3.D.y));
    A = set_point(x,y);
    return A;
}

Bezier3 conversion_bezier2_to_bezier3 (Bezier2 b2)
{
    Bezier3 b3;
    //Point C0
    b3.A = b2.A;

    //Point C3
    b3.D = b2.C;

    double x1,y1,x2,y2;
    //Point C1
    x1 = b2.A.x;
    y1 = b2.A.y;

    x2 = b2.B.x;
    y2 = b2.B.y;

    Point total;
    total = set_point((x1+(2*x2))/3,(y1+(2*y2))/3);
    b3.B = total;

    //Point C2
    x1 = b2.B.x;
    y1 = b2.B.y;

    x2 = b2.C.x;
    y2 = b2.C.y;

    total = set_point(((2*x1)+x2)/3,((2*y1)+y2)/3);
    b3.C = total;

    return b3;
}

Bezier2 approx_bezier2(Contour c, int j1, int j2)
{
    Bezier2 b2;
    int n = j2 - j1;
```



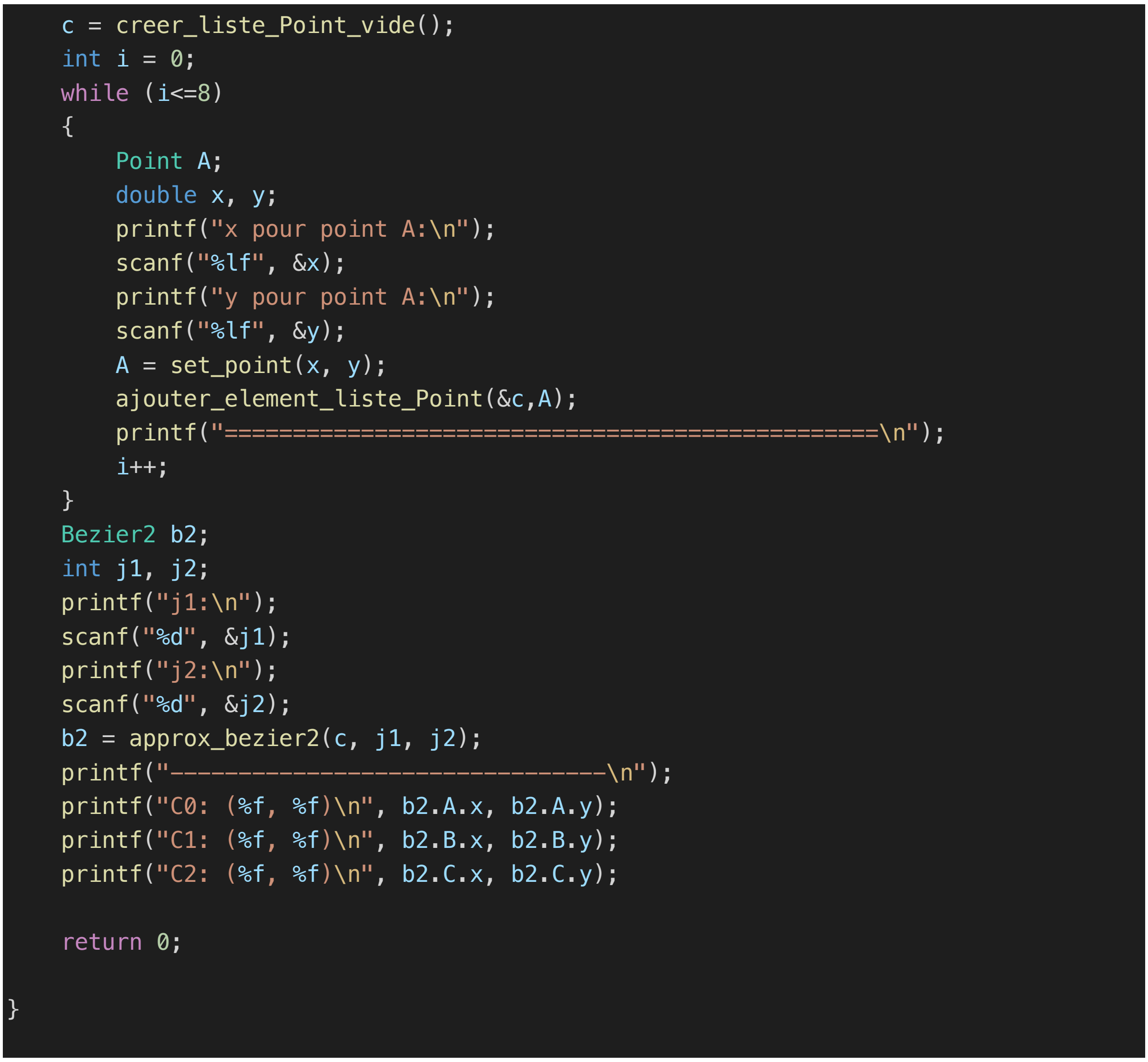
```
    Tableau_Point T = sequence_points_liste_vers_tableau(c);
    Point C0, C2;
    C0 = T.tab[j1];
    C2 = T.tab[j2];

    if (n==1)
    {
        Point C1;
        C1 = set_point((C0.x+C2.x)/2, (C0.y+C2.y)/2);

        //Declaration de la courbe bezier
        b2.A = C0;
        b2.B = C1;
        b2.C = C2;
        return b2;
    }
    else if (n>=2)
    {
        double n_double;
        n_double = (double)(n);

        //Calcul a et b
        double a, b;
        a = (3*n_double)/((n_double*n_double)-1);
        b = ((1-(2*n_double))/(2*(n_double+1)));

        double x=0;
        double y=0;
        Point id;
        for (int i = j1+1; i <j2; i++)
        {
            id = T.tab[i];
            x = x + id.x;
            y = y + id.y;
        }

        //Transformner x et y en double
        double res_x, res_y;


        res_x = a * x + b * ((double)(C0.x)+(double)(C2.x));
        res_y = a * y + b * ((double)(C0.y)+(double)(C2.y));

        Point C1;
        C1 = set_point(res_x, res_y);

        b2.A = C0;
        b2.B = C1;
        b2.C = C2;
        return b2;
    }
```



```
    else
    {
        printf("Error witht the approximation to courbe Bezier2");
        return b2;
    }

}

//FIX THIS
double distance_point_bezier2(Point P1, Bezier2 b2, double ti)
{
    double result;
    Point A;

    A = calcul_ct_bezier2(b2, ti);
    result = distance(P1, A);
    return result;
}


Contour simplification_douglas_peucker_bezier2(Contour C, int j1, int j2, double d)
{
    int n = j2 -j1;

    //Creation de la courbe de Bezier
    Bezier2 b2;
    b2 = approx_bezier2(C, j1, j2);

    Tableau_Point T = sequence_points_liste_vers_tableau(C);



    //Variable initialisations
    double distance, ti;
    double max_distance = 0; //dmax
    int far_away, j;


    for (int i=j1+1; i<j2; i++)
    {
        j = i - j1;
        ti = (double)(j)/(double)(n);
        distance = distance_point_bezier2(T.tab[i], b2, ti);
        if (max_distance < distance)
        {
            max_distance = distance;
            far_away = i;
        }
    }

    if (max_distance <= d)
    {
        Contour L ;
        L = creer_liste_Point_vide();
        ajouter_element_liste_Point(&L, b2.A);
        ajouter_element_liste_Point(&L, b2.B);
        ajouter_element_liste_Point(&L, b2.C);
        return L;
    }
    else
    {

        Contour L1;
        L1 = creer_liste_Point_vide();
        L1 = simplification_douglas_peucker_bezier2(C, j1, far_away, d);

        Contour L2;
        L2 = creer_liste_Point_vide();
        L2 = simplification_douglas_peucker_bezier2(C, far_away, j2, d);

        return concatener_liste_Point(L1, L2);
    }

}


void create_postscript_contours_bezier2(Liste_Contours c, char *file_name, int
hauteur, int largeur) // Mode remplisage uniquement
{
    // Extension managment
    char *no_extension = strtok(file_name, ".");
    char *with_extension = malloc(strlen(no_extension) + 4);
    strcpy(with_extension, no_extension);
    strcat(with_extension, ".eps"); // concantenation

    FILE *fptr;
    fptr = fopen(with_extension, "w");
    if (fptr == NULL)
    {
        printf("EPS File Error!");
        exit(1);
    }

    fprintf(fptr, "%%!PS-Adobe-3.0 EPSF-3.0\n");
    fprintf(fptr, "%%%%BoundingBox:   %d   %d   %d   %d\n", 0, 0, largeur,
hauteur);
    fprintf(fptr, "\n");
    Cellule_Liste_Contours *al;
    al = c.first;
    while (al != NULL)
    {
        Cellule_Liste_Point *el;
        el = (al->data).first;
        Bezier3 b3;
        Bezier2 b2;
        b2.A = el->data;
        el = el->suiv;
        b2.B = el->data;
        el = el->suiv;
        b2.C = el->data;
        b3 = conversion_bezier2_to_bezier3(b2);
        fprintf(fptr, "%.3f %.3f moveto ", b3.A.x, hauteur - b3.A.y);
        fprintf(fptr, "%.3f %.3f %.3f %.3f %.3f %.3f curveto ", b3.B.x, hauteur -
b3.B.y, b3.C.x, hauteur - b3.C.y, b3.D.x, hauteur - b3.D.y);
        el = el->suiv;
        while (el != NULL)
        {
            b2.A = el->data;
            el = el->suiv;
            b2.B = el->data;
            el = el->suiv;
            b2.C = el->data;
            b3 = conversion_bezier2_to_bezier3(b2);
            fprintf(fptr, "%.3f %.3f %.3f %.3f %.3f %.3f curveto ", b3.B.x, hauteur
- b3.B.y, b3.C.x, hauteur - b3.C.y, b3.D.x, hauteur - b3.D.y);
            el = el->suiv;
        }
        fprintf(fptr, "\n 2.0 setlinewidth");
        fprintf(fptr, "\n");
        al = al->suiv;
    }
    fprintf(fptr, "fill\n");
        fprintf(fptr, "\n");
    fprintf(fptr, "\n");
    fprintf(fptr, "showpage\n");
    fclose(fptr);
    return;
}
```

Source code de partie 1.1 (test program : test_approx.c) :

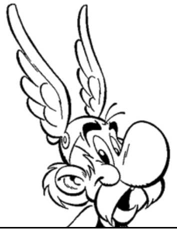
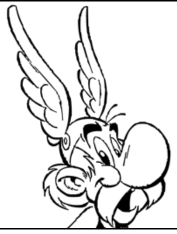
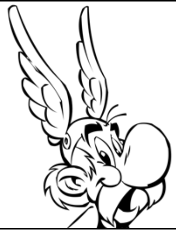
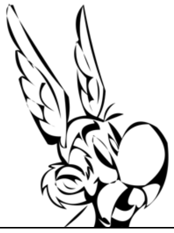
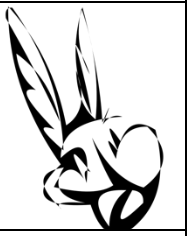
```
#include <stdint.h>
#include <string.h>
#include<stdlib.h>

#include "contour.h"
#include "image.h"

int main(int argc, char **argv)
{
    //Test no 9
    printf("Starting Test 9\n");
    Contour c;
```

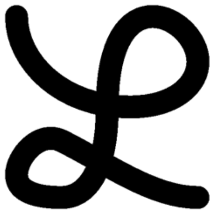
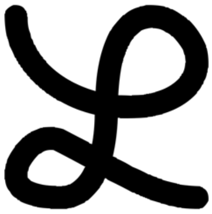
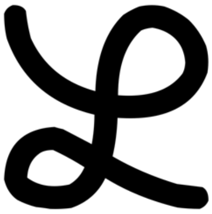
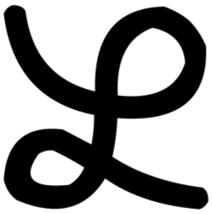
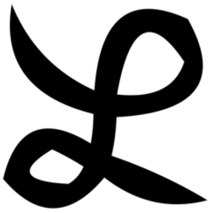
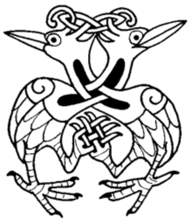
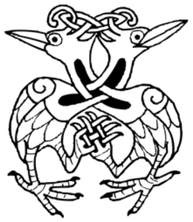
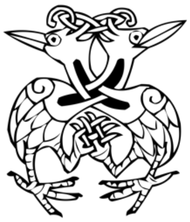
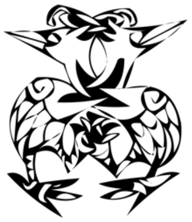
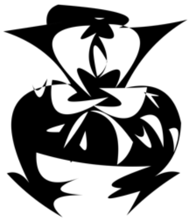


```
    c = creer_liste_Point_vide();
    int i = 0;
    while (i<=8)
    {
        Point A;
        double x, y;
        printf("x pour point A:\n");
        scanf("%lf", &x);
        printf("y pour point A:\n");
        scanf("%lf", &y);
        A = set_point(x, y);
        ajouter_element_liste_Point(&c,A);
        printf("================================================\n");
        i++;
    }
    Bezier2 b2;
    int j1, j2;
    printf("j1:\n");
    scanf("%d", &j1);
    printf("j2:\n");
    scanf("%d", &j2);
    b2 = approx_bezier2(c, j1, j2);
    printf("--------------------------------\n");
    printf("C0: (%f, %f)\n", b2.A.x, b2.A.y);
    printf("C1: (%f, %f)\n", b2.B.x, b2.B.y);
    printf("C2: (%f, %f)\n", b2.C.x, b2.C.y);

    return 0;

}
```

Partie 1.2 :

Table:

Original	D=1	D=3	D=10	D=30
				
Asterix3 Nombre des contours: 32 Nombre des segments totals: 12926	Nombre des bezier totals: 966	Nombre des bezier totals: 296	Nombre des bezier totals: 158	Nombre des bezier totals: 69

				
lettre-L-cursive Nombre des contours: 3 Nombre des segments totals: 4228	Nombre des bezier totals: 255	Nombre des bezier totals: 40	Nombre des bezier totals: 26	Nombre des bezier totals: 15
				
ColombesDeLaPaix Nombre des contours: 106 Nombre des segments totals: 21764	Nombre des bezier totals: 1599	Nombre des bezier totals: 587	Nombre des bezier totals: 295	Nombre des bezier totals: 148