MAT 5932 — Special Topics — Computing for Math

Fall 1993.

Instructor: Bellenot.

The good doctor's Office: 002-B Love, Office Hours: MW 1:30–2:20 or by appointment.

Text: Either Abrahams and Larson "UNIX for the Impatient" or the O'Reilly "Learning the UNIX operating system" by Todino and Strang

Coverage: Many UNIX tools and applications.

Project: Each student will complete an individual project on a topic which most be approved in advance. If time premits these projects will be presented in class.
Homework: There will be weekly homework assignments.
There will be no tests or finals but we may have a quiz or two or three.

Grades: The project is worth 15-20% of your grade. Quizes, if any, will be counted the same as if each was a homework assignment.

```
#include <stdio.h>

int zero = 0;

int factorial1 ( int n )
{
        int i;
        int ans = 1;

        for ( i = 1; i < n; i++ );
        {
                ans = ans * i;
        }
        return ans;
}

int factorial2 ( int n )
{
        if ( n = zero )
        {
                return 1;
        }
        else
        {
                return n * factorial2 ( n - 1 );
        }
}

int factorial3 ( int n )
{
        int answer;

        switch ( n )
        {
        case 0:
                answer = 1;
                break;
        case 1:
                answer = 1;
        case 2:
                answer = 2;
        case 3:
                answer = 6;
        default:
                answer = n * factorial3 ( n - 1 );
        }
        return answer;
}

int factorial4 ( int n )
{
        if ( n < 2 )
                if ( n == 1 )
                        return 1;
        else
                return n * factorial4 ( n - 1 );
}

int factorial5 ( int n )
{
        int answer;

        if ( n > 1 )
                answer = n * factorial5 ( n - 1 );

        return answer;
```

```
}

int factorial6 ( int n )
{
        int temp;
        int ans = 1;

        while ( n )
                temp = n - 1;
                ans = n * factorial6 ( temp );
        return ans;
}

#define LARGE_NUMBER 10
int Cache[LARGE_NUMBER];
int CacheEmpty;

void FillCache ( void )
{
        int i;

        Cache[0] = 1;
        for ( i = 1; i < LARGE_NUMBER; i++ )
                Cache[i] = i * Cache[i - 1];
        CacheEmpty = 0;
}

int factorial7 ( int n )
{
        if ( CacheEmpty )
                FillCache();
        else
                return Cache[n];
}

int factorial8 ( int n )
{
        int answer = 1;

        if ( n == 0 )
                //; answer already correct
        if ( n == 1 )
                ;// answer already correct
        else
                answer = n * factorial8 ( n - 1 );

        return answer;
}

int main ( int argc, char ** argv )
{
        int start = 10;
        int i;

        if ( argc > 1 )
                sscanf ( argv[1], "%d", &start );

        for ( i = start; i; i-- )
                printf ( "Factorial(%d) is %d %d %d %d %d %d %d %d\n",
                        i, factorial1(i), factorial2(i),
                        factorial3(i), factorial4(i), factorial5(i),
                        factorial6(i), factorial7(i), factorial8(i) );
        i = 0;
        printf ( "Factorial(%d) is %d %d %d %d %d %d %d %d\n",
                i, factorial1(i), factorial2(i),
                factorial3(i), factorial4(i), factorial5(i),
```

```
                factorial6(i), factorial7(i), factorial8(i) );
}
```

```
/* coil.c */
#include <math.h>

#define PI M_PI
#define MAX_JUMP 0.025

double theta = PI/6;
double scale = 100;
double x_origin = 2.5;
double y_origin = 2.0;
double t_start = 0;
double t_end = 3;

/* parameteric eqns */

double X(t)
double t;
{
        return cos(2*PI*t);
}

double Y(t)
double t;
{
        return sin(2*PI*t);
}

double Z(t)
double t;
{
        return t;
}

/* transformation 3-space to user space */
transform (x, y, z)
double x, y, z;
{
        double X = y - x * cos( theta );
        double Y = z - x * sin( theta );

        printf ( "%f %f lineto\n", X, Y );
}

start_graph (x, y, z)
double x, y, z;
{
        double X = y - x * cos( theta );
        double Y = z - x * sin( theta );

        printf ( "newpath\n" );
        printf ( "%f %f moveto\n", X, Y );
}

/* draw the parametric curve */

draw ( t_1, x_1, y_1, z_1, t_2, x_2, y_2, z_2 )
double t_1, x_1, y_1, z_1, t_2, x_2, y_2, z_2;
{
        double t_mid = (t_2 + t_1)/2;
        double x_mid = X(t_mid);
        double y_mid = Y(t_mid);
        double z_mid = Z(t_mid);

        double dist_sq = (t_2 - t_1) * (t_2 - t_1) +
                (x_2 - x_mid) * (x_2 - x_mid) +
                (x_1 - x_mid) * (x_1 - x_mid) +
```

```
                (y_2 - y_mid) * (y_2 - y_mid) +
                (y_1 - y_mid) * (y_1 - y_mid) +
                (z_2 - z_mid) * (z_2 - z_mid) +
                (z_1 - z_mid) * (z_1 - z_mid);

        if ( dist_sq < MAX_JUMP)
        {
                transform ( x_2, y_2, z_2 );
        }
        else
        {
                draw ( t_1, x_1, y_1, z_1, t_mid, x_mid, y_mid, z_mid );
                draw ( t_mid, x_mid, y_mid, z_mid, t_2, x_2, y_2, z_2 );
        }
}

/* main routine */
main()
{
        double t_1 = t_start; double t_2 = t_end;
        double x_1 = X(t_1); double y_1 = Y(t_1); double z_1 = Z(t_1);
        double x_2 = X(t_2); double y_2 = Y(t_2); double z_2 = Z(t_2);
        printf ( "%%!-PostScript\n" );
        printf ( "%%%% The Good Doctors Coil.c\n" );
        printf ( "%%%% not EPS\n" );

        printf ( "%f %f scale\n", scale, scale );
        printf ( "%f %f translate\n", x_origin, y_origin );

        printf ( "0 0 moveto\n" );
        transform (1.5, 0.0, 0.0 );
        printf ( "stroke\n" );

        printf ( "0 0 moveto\n" );
        transform (0.0, 1.5, 0.0 );
        printf ( "stroke\n" );

        printf ( "0 0 moveto\n" );
        transform (0.0, 0.0, 3.5 );
        printf ( "stroke\n" );

        start_graph ( x_1, y_1, z_1 );
        draw ( t_1, x_1, y_1, z_1, t_2, x_2, y_2, z_2 );

        printf ( "stroke\nshowpage\n" );
}
```

```
%!-PostScript
%% The Good Doctors Coil.c
%% not EPS
0.0025 setlinewidth
100.000000 100.000000 scale
2.500000 2.000000 translate
0 0 moveto
-1.299038 -0.750000 lineto
stroke
0 0 moveto
1.500000 0.000000 lineto
stroke
0 0 moveto
0.000000 3.500000 lineto
stroke
newpath
-0.866025 -0.500000 moveto
-0.164504 -0.321985 lineto
0.592466 -0.003842 lineto
1.149738 0.378795 lineto
1.319479 0.728553 lineto
1.044475 0.959143 lineto
0.417420 1.024440 lineto
-0.350332 0.934035 lineto
-1.000000 0.750000 lineto
-1.312608 0.565965 lineto
-1.182787 0.475560 lineto
-0.654295 0.540857 lineto
0.094734 0.771447 lineto
0.811832 1.121205 lineto
1.255293 1.503842 lineto
1.275644 1.821985 lineto
0.866025 2.000000 lineto
0.164504 2.009485 lineto
-0.592466 1.878842 lineto
-1.149738 1.683705 lineto
-1.319479 1.521447 lineto
-1.044475 1.478357 lineto
-0.417420 1.600560 lineto
0.350332 1.878465 lineto
1.000000 2.250000 lineto
1.312608 2.621535 lineto
1.182787 2.899440 lineto
0.654295 3.021643 lineto
-0.094734 2.978553 lineto
-0.811832 2.816295 lineto
-1.255293 2.621158 lineto
-1.275644 2.490515 lineto
-0.866025 2.500000 lineto
stroke
showpage
```