Assignment 1: Tasks for Continuous Modelling and Simulation

ODE 1 (Constrained Growth): (1 mark)

$$\frac{dP}{dt} = rP - \left(r\frac{P}{M}\right)P$$

where,

P(0) = 100 is the initial condition,

P is the population,

r = 0.1 is the growth rate, and

M = 1000 is the capacity constraint.

For this ODE, do the following:

1. Create system dynamic diagrams in NetLogo,
2. Implement simulations using Euler Method in C/C++. Your code should use the plot function given in the end of this assignment.

ODE 2 (Newton's Law of Heating and Cooling): (1 mark)

$$\frac{dT}{dt} = hA\left(T - T_{env}\right)$$

where,

T(0) = 25 degree celsius is the initial condition,

T is the Temperature of an object

h is the heat loss coefficient (Watt / metre^2 / Kelvin) (use appropriate value yourself)

A is the surface area of object in metre^2 (use appropriate value yourself)

T_{env} is the temperature of the environment (use appropriate value yourself)

Note: Take care of units

For this ODE, do the following:

1. Create system dynamic diagrams in NetLogo,
2. Implement simulations using Euler Method in C/C++. Your code should use the plot function given in the end of this assignment.

ODE 3 (Gompertz Model for Tumor Growth): (1 mark)

$$\frac{dN}{dt} = kN\ln\left(\frac{M}{N}\right)$$

where,

N(0) = 1000 cells is the initial condition

N is the number of cancer cells

k is the proliferation ability of cancer cells (10% per month)

M = 1 million is the carrying capacity

For this ODE, do the following:

1. Create system dynamic diagrams in NetLogo,
2. Implement simulations using Euler Method in C/C++. Your code should use the plot function given in the end of this assignment.

ODE 4 (1-Compartment Model of Repeated Drug Dose): (1 mark)

$$\frac{dQ_c}{dt} = \frac{1}{V}\frac{dQ}{dt} = k_a d - k_e Q$$

where,

$Q(0) = 0$ is the initial condition

Q is the amount of drug in the compartment

V = 5 Litre is the volume of blood

$Q_c = Q / V$ is the drug concentration in blood

$k_a = 0.12$ is the absorption fraction

d = 300 mg is the drug dosage, taken at 8 hours interval, thrice a day

$h_l = 10$ hours is the half life

$k_e = - \ln(0.5)/ h_l$ is the elimination constant

For this ODE, do the following:

1. Create system dynamic diagrams in NetLogo,
2. Implement simulation using Euler Method in C/C++. Your code should use the plot function (given at end of assignment) to plot Q_c .

ODE 5 (Motion of a Run and Jump): (3 mark)

$$\frac{dP}{dt} = V \quad \text{and} \quad \frac{dV}{dt} = a$$

where,

P is position (with both x, y components)

V is velocity (with both x, y components)

a is the accelaration (a = F / m), where m = mass can be assumed as 1

For this ODE, do the following:

1. See the simple program below. Copy the code and save it in a file. Then compile the code as "gcc <source-code> -lcurses". When you run the code, pressing the left or right arrow of the numeric keypad will make a ball roll across the screen in the said direction. **You have to add the functionality of the above ODE system to the code in order to demonstrate a running and jumping of the ball.** Attempt the implementation of the ODE using Euler's method only (keeping dt = 1)

```
#include <stdio.h>
#include <curses.h>
#define DELAY 60000

int main (int argc, char *argv[])
{    initscr(); noecho();
     curs_set(FALSE);

     int max_x, max_y, i, c;
     getmaxyx(stdscr, max_y, max_x);

     int pos_x = 0, pos_y = max_y - 1, direction_x = 1, next_x;

     WINDOW *menu_win = newwin(max_y, max_x, 0, 0);
     keypad(menu_win, TRUE);

     for (i = 0; i < 5000; i++) {
          clear();
          mvprintw(pos_y, pos_x, "o");
```

```
		refresh();

		c = wgetch(menu_win);
		switch(c)
		{	case KEY_LEFT:		pos_x -= direction_x;	break;
			case KEY_RIGHT:		pos_x += direction_x;	break;
		}
		next_x = pos_x + direction_x;
		if (next_x == max_x)		pos_x = (max_x-2);
		if (next_x == 0)		pos_x = 2;
	}
	endwin();
	return 0;
}
```

ODE 6 (Lorenz Model): (3 mark)

$$\frac{dx}{dt}=\sigma\,(y-x)$$
$$\frac{dy}{dt}=x(\rho-z)-y$$
$$\frac{dz}{dt}=xy-\beta z$$

where,

x(0) = 1, y(0) = 1, z(0) = 1 are the initial conditions

x, y, and z are the positions of a particle in coordinate axis.

σ = 10, ρ = 28, and β = 8/3 are constants (you can play around with them if you want)

For this ODE system, do the following:

1. Implement simulation using Euler Method in C/C++. Set the dt as the maximum dt which can be supported by Euler Method. Use the plot function given at end of assignment to plot the results.
2. Implement simulation using Runge Kutta 4 method in C/C++. Set the dt as the maximum dt which can be supported by Runge-Kutta-4. Use the plot3d function given at end of assignment to plot the results.

Plot Function

```
/*	Code for simple Plot:
	Steps		= Number of Iterations
	dt		= timestep
	x		= Your 1D-array in which all values are stored
*/
void plot(int steps, double dt, double *x)
{
	FILE *gplot = popen("gnuplot -persistent", "w");
	fprintf(gplot, "plot '-' u 1:2 title 'x' with lines\n");
	int i;
	for (i = 0; i <= steps; i++) {
		fprintf(gplot,"%lf %lf\n", i*dt, x[i]*100/3000);
	}
	fprintf(gplot,"e");
}
```

Plot3d Function

```
/*	Code for 3D Plot
```

```
	Steps		= Number of Iterations
	dt		= timestep
	x		= Your 1D-array in which all values of x-axis are stored
	y		= Your 1D-array in which all values of y-axis are stored
	z		= Your 1D-array in which all values of z-axis are stored
*/
void plot3d(int steps, double dt, double *x, double *y, double *z)
{
	FILE *gplot = popen("gnuplot -persistent", "w");
	fprintf(gplot, "splot '-' u 1:2:3 title 'm' with lines\n");
	fprintf(gplot,"%lf %lf %lf\n", x[0], y[0], z[0]);
	int i;
	for (i = 0; i <= steps; i++) {
		fprintf(gplot,"%lf %lf %lf\n", x[i], y[i], z[i]);
	}
	fprintf(gplot,"e");
}
```

Deliverable:

Send me a tar file containing each of the material required. Your file names **MUST** be named as follows:

rollnumber-ode-x-deliverable-y

As an example,

p13-1234-ode-6-deliverable-1.c
p13-1234-ode-6-deliverable-2.c
p13-1234-ode-1-deliverable1.nlogo

etc. etc.