```
#include <stdio.h>
#include <math.h>
// Define symbolic constants
#define MAX_SIZE 500 // maximum size of arrays // Structure definitions
typedef struct
{
   double tempAmbiant; // Ambiant temperature
   double k; // propotionality constant
   double tempInitial; // initial temperature
   double deltat; // time step for Euler method
   double tfinal; // final time for plotting
   int n; // number of values to compute must
   double t[MAX_SIZE]; // time values
   double temp[MAX_SIZE]; // temperature from analytical equation
   double tempEuler[MAX_SIZE]; // temperature from Euler method
} NEWTON;

void getUserInput(NEWTON *);
double getPositiveValue(char *);
double getValue(char *);
void calculateTemperatureAnalytical(NEWTON *);
void calculateTemperatureEuler(NEWTON *);

/*-------------------------------------------
Function: main
Description: Overall control of the program.
Gets the input from the user, calculates temperatures using analytical solution and using Euler's
method, and plot the 2 curves. ----------------------------------------------*/
void main()
{
   NEWTON newton; // Input and output data // Get the user input
   getUserInput(&newton);
   calculateTemperatureAnalytical(&newton);
   calculateTemperatureEuler(&newton);
   printf("All done.");

}
/*---------------------------------------------------------- Function: getUserInput
Parameters:
nPtr - reference to NEWTON structure variable. Members used deltat - delta t variable
tempAmbiant - ambiant temperature
k - proportionality constant
tempInitial - initial temperature
```

```
n - number of elements used in time/velocity arrays Description: Gets from the user values for
ambiant temperature,
proportionality constant, initial temperature,
and time step (delta t) and store in
appropriate variables. Also computes n and ensures that it is less than MAX_SIZE.
-------------------------------------------------------------*/
void getUserInput(NEWTON *nPtr)
{
// Get weight and drag values
   nPtr->tempAmbiant = getValue("the ambiant temperature");
   nPtr->k = getPositiveValue("the proportionalty constant");
   nPtr->tempInitial = getValue("the initial temperature");
// Get time values
   do
   {
      nPtr->tfinal = getPositiveValue("final time");
      nPtr->deltat = getPositiveValue("time step");
      nPtr->n = nPtr->tfinal/nPtr->deltat;
      if(nPtr->n > MAX_SIZE)
         printf("Time step too small for final time (%d)\n", nPtr->n); }

   while(nPtr->n > MAX_SIZE);
}
/*------------------------------------------------------------ Function: getPositiveValue
Parameters:
prompt - reference to string to include in the user prompt Return value: positive value obtained
from the user.
Description: Prompt the user for a value (using the prompt string)
and checks that the value is positive. -------------------------------------------------------------*/
double getPositiveValue(char *prompt)
{
   double value; // Value entered by the user.
   do
   {
      printf("Please enter a value for %s: ", prompt);
      scanf("%lf",&value);
      if(value <= 0.0)
         printf("The value must be greater than zero.\n"); }
   while(value <= 0.0);
   return(value);
}

double getValue(char *prompt)
{
```

```
   double value; // Value entered by the user.
   printf("Please enter a value for %s: ", prompt);
   scanf("%lf",&value);
   return(value);
}

void calculateTemperatureAnalytical(NEWTON *nPtr) {
   double time = 0.0;
   int ix;
   for(ix = 0; ix < nPtr->n; ix = ix + 1) {
      nPtr->t[ix] = time;
      nPtr->temp[ix] = nPtr->tempAmbiant +(nPtr->tempInitial -
nPtr->tempAmbiant)*exp(-nPtr->k*nPtr->t[ix]);
      time = time + nPtr->deltat;
      printf("Analytical Temperature is %f\n",nPtr->temp[ix]);

   } }
/*------------------------------------------------------------ Function: calculateTemperatureEuler Parameters:
nPtr - reference to NEWTON structure variable. Members used tempAmbiant - ambiant
temperature
k - proportionality constant
tempInitial - initial temperature
deltat - delta t variable
n - number of elements used in time/velocity arrays
t - array for saving time values
tempEuler - array for saving temperature values calculated
using Euler's method Description: Fills in the arrays with n points of
time/temperature values using the Euler's method for the
velocity. ------------------------------------------------------------*/
void calculateTemperatureEuler(NEWTON *nPtr)
{
   double time; //time
   int ix; //increment
   double ft; //for computing f(t)
   time = 0.0; //time starts at 0
   nPtr->tempEuler[0] = nPtr->tempInitial;
   for(ix = 1; ix < nPtr->n; ix = ix+1)
   {
      time = time + nPtr->deltat; //increment time for next computation
      ft= -nPtr->k*(nPtr->tempEuler[ix-1]-nPtr->tempAmbiant);
      nPtr->tempEuler[ix] = nPtr->tempEuler[ix-1] + ft*(nPtr->deltat);
      printf("Euler Temperature is %lf\n",nPtr->tempEuler[ix]);
   }
}
```