```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <cmath>
#include <stdlib.h>
#include <TGraph.h>
#include <TCanvas.h>
#include <TAxis.h>

using namespace std;


//Zach Warner
//Project II
//This program calculates and plots the power series P(w) of the sin(wt),harmonic oscillator,pendulum. The time series are transfeed to forouier modes and then back to time series to ensure correctness.


double pi=atan(1)*4.0;
double pi2=atan(1)*8.0;

int main()
{

  // **PART A Variables**
  double N=4000; //number of steps
  double t_ft_graph[4000];
  double y_graph[4000];
  double P_graph[3999], w_graph[3999];
  double t = 0;
  double A[3999],B[3999],P[3999],w[3999],f[3999];
  double temp,tempA,tempB,tempf, tempw;

  // **Part C Variables**
  double xt0,t0,pt0,pt,xt;
  double h=0.0001;// step size RK4
  double x_RK4[5],p_RK4[5];
  double h2;
  double n; // number of steps for RK4
  double x_graph[101],p_graph[101],t_graph[101];
  double p_update,x_update;
  double x[10001];
  double p[10001];
  double kx1,kx2,kx3,kx4,kp1,kp2,kp3,kp4; //k-values to be used in the RK4 metho


  // **Part D Variables **
  double x0,energy0,p0;
```

```
// **PART A**
/*for(int i = 0; i<4000; i++)
  {
    y_graph[i]=sin(t*pi);
    t_graph[i]=t;
    t=t+.01;
    }

for(int i = 0; i<N-1; i++)
  {
    tempA=0;
    tempB=0;
    tempw=(pi2*i)/N;

    for(int k=0; k<N-1; k++)
         {
           temp = tempw*k;
           tempA = tempA + sin(pi*t)*cos(temp);
           tempB = tempB - sin(pi*t)*sin(temp);
           t=t+0.01;
         }

    A[i] = tempA;
    B[i] = tempB;
    P[i] = A[i]*A[i] + B[i]*B[i];
    P_graph[i]=log10(P[i]);
    w_graph[i] = tempw/(0.01*pi2); //where N*0.01 is total time durration (T)

  }

 for(int k = 0; k<N-1; k++)
  {

     tempf = 0;

    for(int n=0; n<N-1; n++)
         {
           temp = ((pi2*k*n)/N);
           tempf = tempf + A[n]*cos(temp)-B[n]*sin(temp);

         }

    f[k]=tempf*(2/N)/1.65;
    }
*/
```

```
// **PART C**

/*xt0 = 0;    // Initial position
  t0 = 0;     // Initial time
  pt0 = 1;    // Initial velocity (momenutm)
  h2 = h*0.5;  // used in RK4 method

  n = 1/h;
  x[0] = 0;//inital conditions for position and momentum
  p[0] = 1;
  x_graph[0]=0;
  p_graph[0]=1;
  t_graph[0]=0;

  for (int i = 0; i < 100 ;i++)
    {
     for(int j=0;j<n;j++)   // RK 4th order method
          {
            kx1=h*p[j];
            kp1=-h*x[j];
            kx2=h*(p[j]+h2*kp1);
            kp2=-h*(x[j]+h2*kx1);
            kx3=h*(p[j]+h2*kp2);
            kp3=-h*(x[j]+h2*kx2);
            kx4=h*(p[j]+h*kp3);
            kp4=-h*(x[j]+h*kx3);
            x[j+1]=(x[j]+(kx1+2*kx2+2*kx3+kx4)/6);
            p[j+1]=(p[j]+(kp1+2*kp2+2*kp3+kp4)/6);
            p_update=p[j+1];
            x_update=x[j+1];
          }
     x_graph[i+1]=x_update;
     p_graph[i+1]=p_update;
     // t_graph[i+1]=i+.1;
     x[0]=x_update;//new initial conditions for RK4 method
     p[0]=p_update;
     }

  for(int i = 0; i<100; i++)
  {
    t_ft_graph[i]=t;
    t=t+1;
    }

  for(int i = 0; i<100; i++)//forward Fourier transform
  {
    tempA=0;
    tempB=0;
```

```
  tempw=(pi2*i)/100;

  for(int k=0; k<100; k++)
       {
        temp = tempw*k;
        tempA = tempA + x_graph[k]*cos(temp);
        tempB = tempB - x_graph[k]*sin(temp);
        t=t+0.01;
       }

  A[i] = tempA;
  B[i] = tempB;
  P[i] = A[i]*A[i] + B[i]*B[i];
  P_graph[i]=log10(P[i]);
  w_graph[i] = tempw/(pi2); //where N*0.01 is total time durration (T)

 }

 for(int k = 0; k<100; k++)//Backward Fourier transform
 {

   tempf = 0;

  for(int n=0; n<100; n++)
       {
        temp = ((pi2*k*n)/100);
        tempf = tempf + A[n]*cos(temp)-B[n]*sin(temp);

       }

  f[k]=tempf*(2./100)/2;

  }*/


//  ** PART D **

 x0 = 0;    // Initial angle
 t0 = 0;     // Initial time
 energy0 = 0;
 p0 = 2;    // Initial velocity (momenutm)
 h2 = h*0.5;  // used in RK4 method

 n = 1/h;
 x[0] = x0;//inital conditions for position and momentum
 p[0] = p0;
 x_graph[0]=0;
 p_graph[0]=p0;
 t_graph[0]=0;
```

```
for (int i = 0; i < 100 ;i++)
  {
   for(int j=0;j<n;j++)   // RK 4th order method
        {
         kx1=h*p[j];
         kp1=-h*(p0*sin(x[j]));
         kx2=h*(p[j]+h2*kp1);
         kp2=-h*(p0*sin(x[j]+h2*kx1));
         kx3=h*(p[j]+h2*kp2);
         kp3=-h*(p0*sin(x[j]+h2*kx2));
         kx4=h*(p[j]+h*kp3);
         kp4=-h*(p0*sin(x[j]+h*kx3));
         x[j+1]=(x[j]+(kx1+2*kx2+2*kx3+kx4)/6);
         p[j+1]=(p[j]+(kp1+2*kp2+2*kp3+kp4)/6);
         p_update=p[j+1];
         x_update=x[j+1];
        }
   x_graph[i+1]=x_update;
   p_graph[i+1]=p_update;
   t_graph[i+1]=i+1;
   x[0]=x_update;//new initial conditions for RK4 method
   p[0]=p_update;
   }

for(int i = 0; i<100; i++)
{
  t_ft_graph[i]=t;
  t=t+1;
  }

for(int i = 0; i<100; i++)//forward Fourier transform
{
  tempA=0;
  tempB=0;
  tempw=(pi2*i)/100;

  for(int k=0; k<100; k++)
       {
         temp = tempw*k;
         tempA = tempA + x_graph[k]*cos(temp);
         tempB = tempB - x_graph[k]*sin(temp);
         t=t+0.01;
       }

  A[i] = tempA;
  B[i] = tempB;
  P[i] = A[i]*A[i] + B[i]*B[i];
  P_graph[i]=log10(P[i]);
```

```
    w_graph[i] = tempw/(pi2); //where N*0.01 is total time durration (T)

   }

   for(int k = 0; k<100; k++)//Backward Fourier transform
   {

     tempf = 0;

    for(int n=0; n<100; n++)
        {
          temp = ((pi2*k*n)/100);
          tempf = tempf + A[n]*cos(temp)-B[n]*sin(temp);

        }

    f[k]=tempf*(2./100)/2;

    }


 //This section is for graphing

  TGraph *gr1 = new TGraph(100,x_graph,p_graph);
  TAxis *axis = gr1->GetXaxis();

  // gr1->Draw("AC");//Draws forward historesis loop
  axis->SetLimits(-2.5,2.5); //x-axis
  gr1->GetHistogram()->SetMaximum(2.5);//y-axis
  gr1->GetHistogram()->SetMinimum(-2.5);
  gr1->GetXaxis()->SetTitle("x");
  gr1->GetYaxis()->SetTitle("P");
  gr1->GetXaxis()->CenterTitle();
  gr1->GetYaxis()->CenterTitle();
  gr1->SetTitle("Phase Space (p=2.0m/s)");
  gr1->SetLineColor(0);
  gr1->SetMarkerColor(4);
  gr1->Draw("AC*");

 return 0;

}
```