http://faculty.virginia.edu/comp-phys/phys2660/wiki/doku.php

```
ssh -Y psa5dg@galileo.phys.virginia.edu

gedit

/*
Name:   Patrick Anderson

UserId: psa5dg

Homework #: 2

Problem #: 1

Program Name: psa5dg_erlang.cpp

Pledge Signature: "On my honor, I pledge that
                        I have neither given nor
                                received help on this assignment."
*/
```

REMEMBER ABOUT THE RANDOM AND HIST LIBRARIES

```
g++ -O -Wall psa5dg_shutthebox.cpp -o box $P2660FLAGS
```

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "hist.hpp"

int main()
{
   FILE *data;
   FILE *bkg;
   FILE *sig;
   char line[80];
   char *ptr;
   double chi2 = 0;
   double chi22 = 0;
   double part1;
   double temp = 100;
   double bind, xd, yd, sigmad;
   double binb, xb, yb, sigmab;
   double bins, xs, ys, sigmas;
   double f1, f2;

   data = fopen("psa5dg_data.dat", "r");
   bkg = fopen("psa5dg_BkgShape.dat", "r");
   sig = fopen("psa5dg_SigShape.dat", "r");

   for (int i=0; i<14; i++) {
      ptr = fgets(line,sizeof(line),data);
      ptr = fgets(line,sizeof(line),bkg);
      ptr = fgets(line,sizeof(line),sig);
   }

   for (int i=0; i<17; i++) {

      ptr = fgets(line,sizeof(line),data);
      fscanf(data, "%lf %lf %lf %lf", &bind, &xd, &yd, &sigmad);

      ptr = fgets(line,sizeof(line),bkg);
      fscanf(bkg, "%lf %lf %lf %lf", &binb, &xb, &yb, &sigmab);
```

```
        ptr = fgets(line,sizeof(line),sig);
        fscanf(sig, "%lf %lf %lf %lf", &bins, &xs, &ys, &sigmas);

        part1 = .5*yb + .5*ys;
        chi2 += ((yd-part1)*(yd-part1))/(sigmad*sigmad);

        for (int i=0; i<100; i++) {
            part1 = ((double)i/100)*yb + ((double)1-((double)i/100))*ys;
            double temp2 = yd-part1;
            if (fabs(temp2) < fabs(temp)) {
                temp = temp2;
                f1 = (double)i/100;
                f2 = (double)1-((double)i/100);
            }
        }
        printf("fb: %lf      fs: %lf\n", f1, f2);
        chi22 += (temp*temp)/(sigmad*sigmad);

    }
    fclose(data);
    fclose(bkg);
    fclose(sig);

    printf("chi^2 with a 50/50 mix: %lf\n", chi2);
    printf("chi^2 with a more precise mix: %lf\n", chi22);

    return 0;
}

    h1 hist;
    h1init(&hist, 50, 0, 1000, "Ht Histogram");
    h1labels(&hist, "values", "nParticles");
    FILE *inp;
    inp = fopen("data.dat", "r");
    char line[80];
    char *ptr;
    double one, two, three, four, five, six, seven, ht, nine;
    while (1) {
        ptr = fgets(line,sizeof(line),inp);
        if (feof(inp)) break;
        if (fscanf(inp, "%lf %lf %lf %lf %lf %lf %lf %lf %lf",&one,&two,&three,&four,&five,&six,&seven,&ht,&nine)
== 9) {
            h1fill(&hist, ht);
        }

    }
    fclose(inp);
    h1dump(&hist, "psa5dg_data.dat");
    h1plot(&hist,"");

/*
Name:   Patrick Anderson

UserId: psa5dg

Homework #: Final Exam

Problem #: 2

Program Name: psa5dg_blackjack.cpp
```

```
Pledge Signature: "On my honor, I pledge that
                        I have neither given nor
                                received help on this assignment."
*/

/*
If the player has a score of 15 and the dealer has a score of 6,
the player should stand. Standing has a 42% chance of not losing,
as opposed to a 34% chance of not losing if he or she hits.
*/

/*
  For a player score of 15:
Dealer's Card: 2 S
Dealer's Card: 3 S
Dealer's Card: 4 S
Dealer's Card: 5 S
Dealer's Card: 6 S
Dealer's Card: 7 H
Dealer's Card: 8 H
Dealer's Card: 9 H
Dealer's Card: 10   H
Dealer's Card: A H
*/

/*
   Player's Score:  8   9   10  11  12  13  14  15  16
Dealer's Card: 2 H   H   H   H   H   H   S   S   S
Dealer's Card: 3 H   H   H   H   H   H   S   S   S
Dealer's Card: 4 H   H   H   H   H   S   S   S   S
Dealer's Card: 5 H   H   H   H   H   S   S   S   S
Dealer's Card: 6 H   H   H   H   H   S   S   S   S
Dealer's Card: 7 H   H   H   H   H   H   H   H   H
Dealer's Card: 8 H   H   H   H   H   H   H   H   H
Dealer's Card: 9 H   H   H   H   H   H   H   H   H
Dealer's Card: 10   H   H   H   H   H   H   H   H   H
Dealer's Card: A H   H   H   H   H   H   H   H   H
*/

#include <stdio.h>
#include <stdlib.h>
#include "random.hpp"
#include "hist.hpp"

char searchFunction(int player, int computer);

int main(int argc, char *argv[])
{
   int nGames;
   int player;
   int computer;
   int deck[4][13];
   int card;
   int acep;
   int acec;
   char option;
   char breaker;

   if (strcmp(argv[2],"all") == 0) player = 0;
   else player = atoi(argv[2]);
   if (strcmp(argv[3],"all") == 0) computer = 0;
   else computer = atoi(argv[3]);
   breaker = searchFunction(player, computer);
   if (breaker == 'x') return 0;
```

```c
for (int i=0; i<4; i++) {
    for (int j=0; j<13; j++) {
        if (j==0) deck[i][j] = 11;
        else if (j<10) deck[i][j] = j+1;
        else deck[i][j] = 10;
    }
}
nGames = atoi(argv[1]);

for (int i=0; i<nGames; i++)
{
    acep = 0;
    acec = 0;
    option = 'x';
    if (strcmp(argv[2],"A") == 0)
    {
        player = 11;
        acep = 1;
    }
    else player = atoi(argv[2]);
    if (strcmp(argv[3],"A") == 0)
    {
        computer = 11;
        acec = 1;
    }
    else computer = atoi(argv[3]);
    while (option != 's') {
        printf("\nPlayer's Score: %d\n", player);
        printf("Dealer's Face-Up Card: %d\n", computer);
        printf("Recommendation: %c\n", searchFunction(player,computer));
        printf("\nHit or Stand? ('enter' to hit, 's' to stand): ");
        scanf("%c", &option);
        if (option != 's')
        {
            card = deck[randui(0,3)][randui(0, 12)];
            if (card == 11) acep++;
            player += card;
        if (player > 21 && acep>0)
        {
            player -= 10;
            acep--;
        }
        }
        if (player > 21)
        {
            printf("Player Busted!\n");
            break;
        }
    }
    do  {
        card = deck[randui(0,3)][randui(0, 12)];
        if (card == 11) acec++;
        computer += card;
        if (computer > 21 && acec>0)
        {
            computer -= 10;
            acec--;
        }
    } while (computer < 17);
    printf("Player's Score: %d\n", player);
    printf("Dealer's Score: %d\n", computer);
    if ((player < 22 && player >= computer) || (player < 22 && computer > 21))
    {
            printf("You didn't lose!\n");
        }
        else printf("You lost!\n");
    }

    return 0;
}

char searchFunction(int player, int computer)
{
    char options[15][11];
    for (int i=0; i<15; i++) {
        for (int j=0; j<11; j++) {
            if (i<5) options[i][j] = 'H';
            else if (i>8) options[i][j] = 'S';
            else if ((i>5 && i<9) && j<5) options[i][j] = 'S';
            else if (i==5 && (j<5 && j>1)) options[i][j] = 'S';
            else options[i][j] = 'H';
        }
    }

    if (player == 0 && computer == 0)
    {
        printf("Columns: Player's Score from 8-21:\n");
        printf("Rows: Dealer's Card from 2-A:\n");
        for (int i=0; i<10; i++)
        {
        for (int j=0; j<14; j++)
        {
            printf("%2c     ", options[j][i]);
        }
        printf("\n\n");
        }
        return 'x';
    }
    if (player == 0)
    {
        for (int i=0; i<14; i++) printf("%d:  %c\n", i+8, options[i][computer-2]);
        return 'x';
    }
    if (computer == 0)
    {
        for (int i=0; i<10; i++) printf("%d:  %c\n", i+2, options[player-8][i]);
        return 'x';
    }

    return options[player-8][computer-2];
}


#include <stdio.h>
#include <stdlib.h>
#include <string.h>

const char *password = NULL;

void fileReader(char inputName[50], char outputName[50]);
void fileReader(char inputName[50]);

int main(int argc, char *argv[])
{
    if (argc != 4 || strlen(argv[1]) < 1 || strlen(argv[2]) < 1 || strlen(argv[3]) < 1)
    {
        printf("Error! Input the password, then the name of the input file, then the
name of the output file or \"-\"!\n");
        return 1;
    }

    password = argv[1];

    if ((strcmp(argv[3], "-")) == 0) fileReader(argv[2]);
    else fileReader(argv[2], argv[3]);

    return 0;
}

void fileReader(char inputName[50], char outputName[50])
{
    FILE *infile;
    FILE *outfile;
    int passCount = 0;
    int passLength = strlen(password);

    infile = fopen(inputName,"rb");
    outfile = fopen(outputName,"wb");

    while (1)
    {
        char c = (char) fgetc(infile); // Read a byte from infile.
        if (feof(infile)) break; // If we've hit the end of the file, quit reading.
        if (passCount == passLength) passCount = 0;
        char d = c^password[passCount];
        fputc(d,outfile);
        passCount++;
    }

    fclose(infile);
    fclose(outfile);
}

void fileReader(char inputName[50])
{
    FILE *infile;
    int passCount = 0;
    int passLength = strlen(password);

    infile = fopen(inputName,"rb");

    while (1)
    {
        char c = (char) fgetc(infile); // Read a byte from infile.
        if (feof(infile)) break; // If we've hit the end of the file, quit reading.
        if (passCount == passLength) passCount = 0;
        char d = c^password[passCount];
        printf("%c", d);
        passCount++;
    }

    fclose(infile);
}
```

```
// Rather than using a static variable to determine the
// ymax of f(x), instead, we could make our random
// number for the y-axis between 0 and f(x), with x
// being the random x value we previously found within the
// given range.

#include <stdio.h>
#include <math.h>
#include "random.hpp"
#include "hist.hpp"

float ymax(float (*f)(float), float xmin, float xmax);
float eFunction(float x);
float denomFunction(float x);
float cosFunction(float x);
void eRandMaker(float xmin, float xmax, float ymax);
void denomRandMaker(float xmin, float xmax, float ymax);
void cosRandMaker(float xmin, float xmax, float ymax);

const float PI = 3.1416;
const float EYMAX = 1.05*ymax(eFunction, -3, 3);
const float DENOMYMAX = 1.05*ymax(denomFunction, 60, 140);
const float COSYMAX = 1.05*ymax(cosFunction, -10, 10);

int main()
{
    eRandMaker(-3, 3, EYMAX);
    denomRandMaker(60, 140, DENOMYMAX);
    cosRandMaker(-10, 10, COSYMAX);
    return 0;
}

float ymax(float (*f)(float), float xmin, float xmax)
{
    float check1 = 0;
    float check2 = 0;
    float steps = 100;
    float dx = (xmax-xmin)/steps;
    for (float i=xmin; i<=xmax; i+=dx)
    {
        check2 = f(i);
        if (check2 > check1) check1 = check2;
    }
    return check1;
}

float eFunction(float x)
{
    return exp(-(x*x));
}
float denomFunction(float x)
{
    return 10/((2*PI)*(((x-100)*(x-100))+25));
}
float cosFunction(float x)
{
    return (cos(x)*cos(x))*exp(-((x*x)/25));
}

void eRandMaker(float xmin, float xmax, float ymax)
{
    h1 hist;
    h1init(&hist, 100, -3, 3, "Function 1");
    h1labels(&hist, "X", "Y");
```

```
	for (int i=0; i<10000; i++)
	{
		float xrand = randu(xmin, xmax);
		float yrand = randu(0, ymax);
		float yfunc = eFunction(xrand);
		if (yrand < yfunc) h1fill(&hist, xrand);
	}
	h1errors(&hist, 1);
	h1plot(&hist, "");
}
void denomRandMaker(float xmin, float xmax, float ymax)
{
	h1 hist;
	h1init(&hist, 100, 60, 140, "Function 2");
	h1labels(&hist, "X", "Y");

	for (int i=0; i<10000; i++)
	{
		float xrand = randu(xmin, xmax);
		float yrand = randu(0, ymax);
		float yfunc = denomFunction(xrand);
		if (yrand < yfunc) h1fill(&hist, xrand);
	}
	h1errors(&hist, 1);
	h1plot(&hist, "");
}
void cosRandMaker(float xmin, float xmax, float ymax)
{
	h1 hist;
	h1init(&hist, 100, -10, 10, "Function 3");
	h1labels(&hist, "X", "Y");

	for (int i=0; i<10000; i++)
	{
		float xrand = randu(xmin, xmax);
		float yrand = randu(0, ymax);
		float yfunc = cosFunction(xrand);
		if (yrand < yfunc) h1fill(&hist, xrand);
	}
	h1errors(&hist, 1);
	h1plot(&hist, "");
}


#include <stdio.h>
#include <stdlib.h>
#include "random.hpp"
#include "hist.hpp"

int mosquito(int distance);
void plotter(int distance);

int main(int argc, char *argv[])
{
	if (argv[1]==NULL || !atoi(argv[1]) || argv[2]!=NULL)
	{
		printf("Error! Input one integer for the number of meters walked into the command line!\n");
		return 1;
	}
	plotter(atoi(argv[1]));

	return 0;
}

void plotter(int distance)
{
	h1 hist;
	h1init(&hist, 60, .5, 60.5, "Mosquito Bites"); //Change values for each distance
	h1labels(&hist, "Number of Bites", "Number of Trials");

	for (int j=0; j<500; j++)
	{
		h1fill(&hist, mosquito(distance));
	}

	h1plot(&hist, "");
	h1plot(&hist, "psa5dg_bite2500.pdf"); //Change name for each distance
}

int mosquito(int distance)
{
	int count = 0;
	for (int i=0; i<distance; i++)
	{
		for (int j=0; j<100; j++)
		{
			int rand = randui(0, 9999);
			if (rand == 0) count++;
		}
	}
	return count;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

const int MAX_NUM = 100;
const double G = 6.673e-11;

typedef struct{
  double s_vec[3];  //- a position coordinate (x,y,z) (stored as a 3 component array)
  double v_vec[3];  //- a velocity vector (vx,vy,vz) (stored as a 3 component array)
  double f_vec[3];  //- a force vector (fx,fy,fz) (stored as a 3 component array)
  double mass; //- a mass, m [kg]
} body;

int fileReader(char fileName[20], body *spheres);
int centerMass(body *spheres, int count);
int forceVectors(body *spheres, int count);

int main(int argc, char *argv[])
{
    char inputter[20];
    body spheres[MAX_NUM];
    int count = 0;
    if (argv[1]==NULL)
    {
        printf("Error! File not found!\n");
        return 1;
    }
    strcpy(inputter, argv[1]);
    count = fileReader(inputter, spheres);
    centerMass(spheres, count);
    forceVectors(spheres, count);
    return 0;
}

int fileReader(char fileName[20], body *spheres)
{
    FILE *input;
    input = fopen(fileName, "r");
    int status;
    int count=0;
    for (count=0; count<MAX_NUM; count++)
    {
        status=fscanf(input, "%lf %lf %lf %lf %lf %lf %lf", &(spheres[count].s_vec[0]),
&(spheres[count].s_vec[1]), &(spheres[count].s_vec[2]), &(spheres[count].v_vec[0]),
&(spheres[count].v_vec[1]), &(spheres[count].v_vec[2]), &(spheres[count].mass));
        if (status==EOF || status!=7) break;
    }
    fclose(input);
    return count;
}

int centerMass(body *spheres, int count)
{
    double xer=0, yer=0, zer=0;
    double den=0;
    for (int i=0; i<count; i++)
    {
        xer += (spheres[i].s_vec[0])*(spheres[i].mass);
        yer += (spheres[i].s_vec[1])*(spheres[i].mass);
        zer += (spheres[i].s_vec[2])*(spheres[i].mass);
        den += (spheres[i].mass);
    }
    printf("\nCenter of Mass: (%4.4lf, %4.4lf, %4.4lf)\n\n", xer/den, yer/den, zer/den);
    return 0;
}

int forceVectors(body *spheres, int count)
{
    double Fx=0, Fy=0, Fz=0;
    double num = 0;
    double den = 0;
    double distance = 0;
    double topx = 0, topy = 0, topz = 0;
    double force = 0;
    for (int i=0; i<count; i++)
    {
        for (int j=0; j<count; j++)
        {
            if (j!=i)
            {
                num = G*(spheres[i].mass)*(spheres[j].mass);

                distance = sqrt((((spheres[j].s_vec[0])-
(spheres[i].s_vec[0]))*((spheres[j].s_vec[0])-(spheres[i].s_vec[0])))+(((spheres[j].s_vec[1])-
(spheres[i].s_vec[1]))*((spheres[j].s_vec[1])-(spheres[i].s_vec[1])))+(((spheres[j].s_vec[2])-
(spheres[i].s_vec[2]))*((spheres[j].s_vec[2])-(spheres[i].s_vec[2]))));

                den = distance*distance;

                force = num/den;

                topx = ((spheres[j].s_vec[0])-(spheres[i].s_vec[0]))/distance;
                topy = ((spheres[j].s_vec[1])-(spheres[i].s_vec[1]))/distance;
                topz = ((spheres[j].s_vec[2])-(spheres[i].s_vec[2]))/distance;

                Fx+=(force*topx);
                Fy+=(force*topy);
                Fz+=(force*topz);
            }
        }
        printf("Body %3d Force = (%8.2lg,%8.2lg,%8.2lg)\n",i+1, Fx, Fy, Fz);
    }
```

```c
    printf("\n");
    return 0;
}



#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(int argc, char *argv[])
{
    float temp1 = atof(argv[1]);
    float temp2 = atof(argv[2]);
    float temp3 = atof(argv[3]);
    float temp4 = atof(argv[4]);
    int rows = 10;
    int columns = 10;
    double sumOld = 0;
    double sumNew = 0;

    printf("How many rows? (including the 4 corner nodes): ");
    scanf("%d", &rows);
    printf("How many columns? (including the 4 corner nodes): ");
    scanf("%d", &columns);
    printf("\n");

    float nodeArray[rows][columns];

    for (int i=0; i<rows; i++)
    {
        for (int j=0; j<columns; j++)
        {
            nodeArray[i][j] = 0;

            nodeArray[0][j] = temp1;
            nodeArray[i][columns-1] = temp2;
            nodeArray[rows-1][j] = temp3;
            nodeArray[i][0] = temp4;

        }
    }

    do
    {
        sumNew = sumOld;
        sumOld = 0;

        for (int i=1; i<rows-1; i++)
        {
            for (int j=1; j<columns-1; j++)
            {
                nodeArray[i][j] = (nodeArray[i-1][j] + nodeArray[i+1][j] + nodeArray[i][j-1] +
nodeArray[i][j+1])/4;
                sumOld += nodeArray[i][j];
            }
        }
    } while ((sumOld-sumNew > .0001) || (sumOld-sumNew < -.0001));

    for (int i=0; i<rows; i++)
    {
        for (int j=0; j<columns; j++)
        {
            if ((i==0 && j==0) || (i==0 && j==columns-1) || (i==rows-1 && j==0) || (i==rows-1 &&
j==columns-1))
            {
                printf("           ");
            }
            else printf("%2.3f     ", nodeArray[i][j]);
        }
        printf("\n\n");
    }

    return 0;
```

```
}
```

```
#include <stdio.h>
#include <math.h>

double trap_rule2(double (*f)(double,double), double xmin, double xmax, double ymin, double ymax, int
steps);
double useFunction(double x, double y);

int main(void)
{
  printf("The volume under the function is %lf\n", trap_rule2(useFunction, 0, 10, 10, 20, 100));
  //printf("%lf\n", useFunction(10, 10));

  return 0;
}

double trap_rule2(double (*f)(double,double), double xmin, double xmax, double ymin, double ymax, int
steps)
{
  double dx = (xmax-xmin)/steps;
  double dy = (ymax-ymin)/steps;
  double xsum=0, ysum=0;
  double i,j;
  double xres=0;
  for (i = ymin; i <= ymax; i += dy)
    {
      for (j = 1; j < steps; j++)
    {
      xsum += f(xmin+j*dx,i);
      xres = (f(xmin, i)+f(xmax, i))*dx/2.+xsum*dx;
    }
      ysum = xres*dy;
    }
  return ysum;
}

double useFunction(double x, double y)
{
  return exp(cos(y))*exp(sin(x+y));
}
```

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 10000

float movingAvg (float noisies[], int nPoints);

int main(int argc, char *argv[])
```

```
{
  int nPoints = atoi(argv[1]);
  float noisies[MAX];
  int n=0;
  int  status;
  FILE *input;

  input = fopen("noisy.dat", "r");
  while (1)
    {
      status = fscanf(input, "%d %f", &n, &noisies[n]);
      if (status == EOF) break;
      n++;
    }
  fclose(input);
  movingAvg(noisies, nPoints);
  return 0;
}

float movingAvg (float noisies[], int nPoints)
{
  double sum = 0;
  int i,j;
  for (i = 0; i < MAX-nPoints; i++)
    {
      for (j = i; j < nPoints+i; j++)
    {
      sum += noisies[j];
    }
      sum /= nPoints;
      printf("%d %20.8lf\n", j-1, sum);
      sum = 0;
    }

  return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#define PI 3.1415927

void hypersphere(double *x, double *y, double *z, double *t);
double standardDeviation(double sum, double sumOfSquares);

int main() {
  double incount=0;
  double hypervolume=0;
  double x,y,z,t;
  double hyperAvg=0;
  double sumOfSquares=0;

  srand((unsigned)time(NULL));
  for(int i=0; i<1000; i++) {


    for(int j=0; j<5000; j++) {
      hypersphere(&x,&y,&z,&t);
      if ((x*x+y*y+z*z+t*t)<1.0) incount++;
    }


    hypervolume = (double)incount/(double)5000*16.0;
    incount = 0;
    hyperAvg += hypervolume;
    sumOfSquares += (hypervolume*hypervolume);
  }


  double sD = standardDeviation(hyperAvg, sumOfSquares);
  hyperAvg /= 1000;
  printf("My calculation is: %lf\n",hyperAvg);
  printf("The standard deviation is: %lf\n",sD);
  return 0;
}



void hypersphere(double *x, double *y, double *z, double *t) {
  *x = (double)rand()/(double)RAND_MAX*2.0-1.0;
  *y = (double)rand()/(double)RAND_MAX*2.0-1.0;
  *z = (double)rand()/(double)RAND_MAX*2.0-1.0;
  *t = (double)rand()/(double)RAND_MAX*2.0-1.0;
}



double standardDeviation(double sum, double sumOfSquares) {
  double sigma = sum*sum;
  sigma = sigma/1000.0;
  double brackets = sumOfSquares - sigma;
  double penultimate = brackets/999.0;
  return sqrt(penultimate);
  }
```

```
#include <stdio.h>
#include <math.h>

double series(double n,double p) {
  return pow((1/n),p);
}

int main(void) {

  double p;
  double newFinal=0;
  double oldFinal=0;
  double n=1;

  printf("Enter a number greater than 1.\n");
  scanf("%lf",&p);

  do {
    oldFinal=newFinal;
    newFinal+=series(n,p);
    n++;
  } while (newFinal-oldFinal>.00000001);
  printf("The final value calculated after evaluating the series is %lf.\n",newFinal);
  return 0;
}
```

```
#include <stdio.h>
#include <math.h>
#define A_0 1.0

double undecayed(double t) {
  return A_0 * pow(.5, t);
        }
 int main(void) {
   FILE *outp;
   outp = fopen("decay.dat", "w");
    printf("#t/t_half Fraction remaining\n");
    fprintf(outp, "#t/t_half Fraction remaining\n");
    for (double i=0; i<10; i++) {
      printf("%.2lf     %.3lf\n", i, undecayed(i));
      fprintf(outp, "%.2lf     %.3lf\n", i, undecayed(i));
             }
    printf("%.1lf     %.3lf\n", 10.0, undecayed(10));
    fprintf(outp, "%.1lf     %.3lf\n", 10.0, undecayed(10));
    fclose(outp);
  }

"legendre.dat" using 1:7 with lines,


 FILE *inp, *outp;
  inp = fopen("young.dat", "r");
  outp = fopen("young.out", "w");

  if (inp==NULL) {
    printf("Error: young.dat not found!\n");
    return 1;
  }


  double length = 0;
  double radius = 0;

  printf("What is the length of the object in meters?\n");
  scanf("%lf",&length);
  printf("What is the radius of the object centimeters?\n");
  scanf("%lf",&radius);


  char *line_ptr;
  char one_line[80];
  int counter = 5;

  char material[20];
  double modulus;
  double limit;
  double delta1N=0;
  double delta100N=0;
  double deltaMax=0;

  while (counter > 0) {
    line_ptr=fgets(one_line,sizeof(one_line),inp);
    if (fscanf(inp, "%s %lf %lf",material,&modulus,&limit) == 3) {
      modulus = modulus*1E10;
      limit = limit*1E8;
      delta1N = (1/modulus)*(length/(radius*radius*PI));
      delta100N = (100/modulus)*(length/(radius*radius*PI));
      deltaMax = limit*length/modulus;
      printf("%s %e %e %e %e %lf\n",material, modulus, limit, delta1N, delta100N, deltaMax);
      fprintf(outp, "%s %e %e %e %e %lf\n",material, modulus, limit, delta1N, delta100N, deltaMax);
      counter--;
    }
```