

Title: Elemental Monte Carlo Methods in Computer Science

Brief Overview:

This unit will introduce computer science students to the use of Monte Carlo methods to evaluate mathematical problems and models. Statistical simulations will be used to determine the value of π , to integrate a continuous function, and to determine probabilities. C++ will be the programming language referred to throughout this unit, but it could be adapted to use other programming languages or spreadsheets. Alternative solutions using a spreadsheet are provided for two of the labs.

Links to NCTM Standards:

- **Mathematics as Problem Solving**
Students will apply mathematical modeling to real-world situations.
- **Mathematics as Communication**
Students will use mathematical terms to effectively communicate their results.
- **Mathematical Connections**
Students will recognize equivalent representations of a problem.
- **Statistics**
Students will design a statistical experiment to study a problem, conduct the experiment via a C++ program, and interpret and communicate the outcomes.
- **Probability**
Students will use simulations to estimate the probability of an event occurring.

Grade/Level:

Grades 10-12

Duration/Length:

This activity will require five regular class periods or two and one-half block periods.

Prerequisite Knowledge:

Students should have working knowledge of the following skills:

- Determination of percentages
- Calculation of area of simple geometric shapes
- Construction of C++ loop structures
- The use of vectors or arrays

Objectives:

Students will:

- learn to use the random function.
- use random numbers to statistically determine the area within a circle and under a curve.
- determine the probability of two people in a group having the same birthday.
- gain an understanding of Monte Carlo techniques and their use in real-world situations.

Materials/Resources/Printed Materials:

- Computer with C++ compiler
- Accompanying Lab and Assignment Sheets

Development/Procedures:

- Using Teacher Note 1, the teacher will introduce students to basic Monte Carlo techniques for determining the area under a curve by using random numbers. The basic concept employed is to randomly cover a rectangle enclosing the function (or figure) with points and determining the ratio of points on or below the function (or points on or within the figure) to the total number of points.
- Using Lab 1, students will verify the hypothesized method of determining area working first with simple rectangle, then a circle, and, finally, a more complex function.
- Assign project as a follow-on activity. Project results may be used to support enrichment lab exercise.
- Using Lab 2, students will use the Monte Carlo method learned in Lab 1 to determine how high an experimental rocket will be at shut down with specified factors.
- Using Teacher Note 2, the teacher will introduce students to other uses of Monte Carlo techniques, e.g., the determination of probabilities.
- Using Lab 3, the students will develop an algorithm and program to determine the probability of two people in groups of 20, 30, or 40 having the same birthday.

Evaluation:

This section will be performance-based (a performance assessment task), in accordance with the move to performance-based assessment. See assessment rubric.

Extension/Follow Up:

- Have the students choose an application problem from their project report for which they could write an algorithm and program using Monte Carlo methods.

Authors:

Tracy A. Birell
Mount Vernon High School
Alexandria, VA

Charles W. Brewer
Lake Braddock Secondary School
Burke, VA

Elemental Monte Carlo Methods in Computer Science

Teacher Note 1

Using Overhead 1, introduce Monte Carlo methods using the following sequence:

- Have students determine the area of the large rectangle. (*6 square units*)
- Have the students determine the area of the shaded rectangle. (*4 units*)
- Have the students determine the ratio between the two rectangles. ($\frac{4}{6}$)
- Pose the following question:
“If you toss 100 darts at the large rectangle and the darts are equally likely to land on any spot in the rectangle, how many would you expect to land in the shaded rectangular area?” (approximately 67, i.e., two-thirds of 100)

Prepare students for first lab exercise by:

- Asking students how the darts might be simulated in a computer program. (Alternatively, a spreadsheet can be used instead of a C++ program. See Lab 1A.)
- Introducing students to the random number generator and randomize functions. Discuss the randomize function, randomization of integers, and randomization of doubles.
- Having students complete Lab 1 using Shell 1. Overhead 2 can be used to discuss the second part of Lab 1.

Assign research project (Project Sheet) to increase student understanding of use of Monte Carlo techniques and as preparation for possible enrichment lab activity

- Indicate to students that they should use both Internet and library resources in their research.
- Recommend students be given one week to complete this assignment.

Prepare students for Lab 2 by:

- Having students read lab sheet, answer questions, and develop algorithm for computer model as homework
- In class, review algorithms before having students develop computer program (or spreadsheet) to solve the problem.

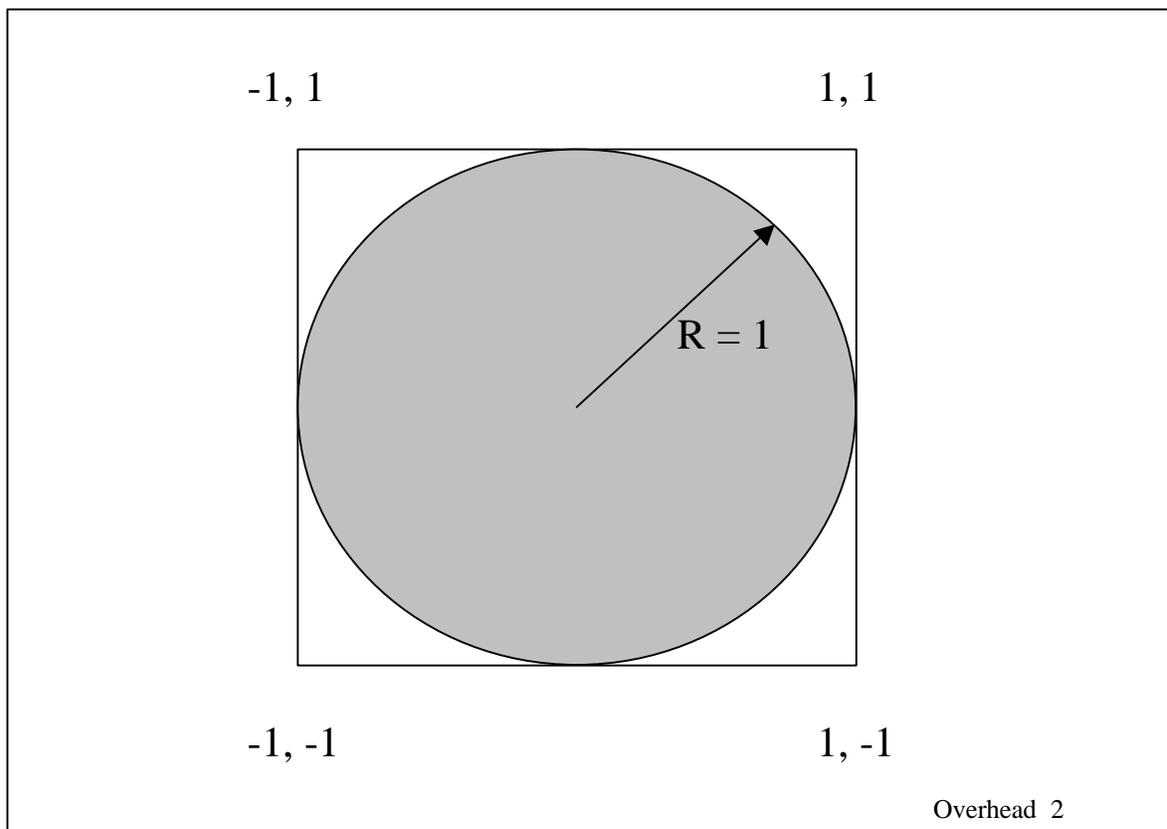
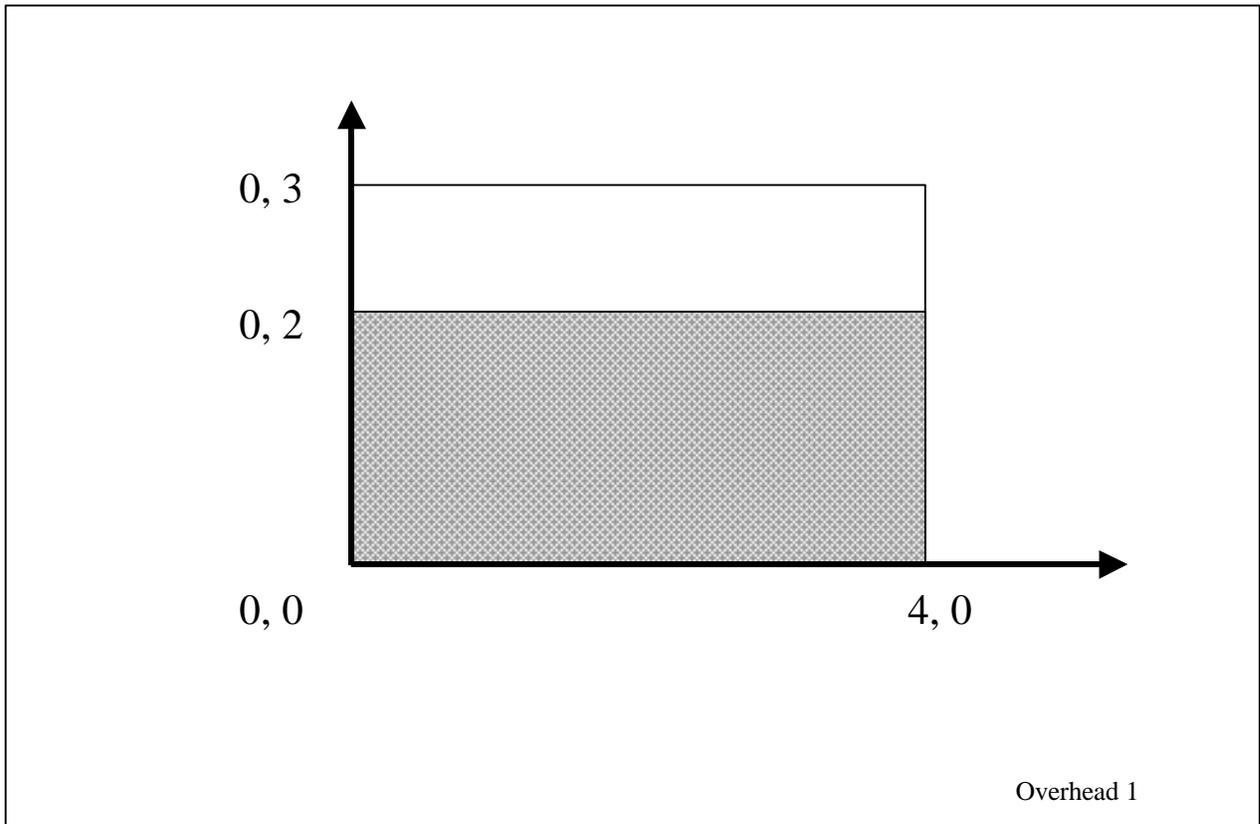
Elemental Monte Carlo Methods in Computer Science

Teacher Note 2

In the first part of this unit, random numbers were used to estimate the area under a curve. In this part, random numbers will be used to determine the probability of an event occurring. The question to be considered and programmed is one that is often discussed in math classes when discussing probabilities, specifically, “in a group of a given size, what is the likelihood of two people having the same birthday?”

- To introduce the problem, the teacher may pose the question: “Do you think two people in the class have the same birthday?” And then follow-up with a check on birthdays. (For a class of thirty, the probability is approximately 70 percent.)
- Ask students how they might determine an estimate of this probability using the tools that have been explored in the previous labs.
 - One technique is to use a loop within a loop and the random function. First, create an apvector (or array) of counters of length 365. Within the inner loop, use the random function to determine the index of the apvector and then increment that element. After the desired number of loop iterations (the number of people in the group), search the apvector for any element that is greater than 1. Increment a counter for each occurrence. Repeat the outer loop multiple times to estimate the probability. The probability is the total occurrences divided by the number of tries.
 - A second method is similar to the first except that the apvector would represent people and the random function would be used to generate a birthday for each person (element). The search function would then have to compare elements to find any matches and increment a counter for each match. (This method will require more code as a loop within a loop will be required for this search.) As in the first method, repeat multiple times to determine the probability.

Elemental Monte Carlo Methods in Computer Science - OVERHEADS



**Elemental Monte Carlo Methods
In Computer Science
Project**

Name_____

Date_____

Prologue: In many fields, e.g. medical research, insurance, physics, etc., Monte Carlo techniques are used to determine solutions to problems that are resistive to more conventional mathematical analysis.

Directions: Using the Internet and library resources, research Monte Carlo techniques. Find five examples of where Monte Carlo techniques are used in business, research, or other areas. Briefly describe (in three to five sentences each) how Monte Carlo techniques are used in each example.

Elemental Monte Carlo Methods

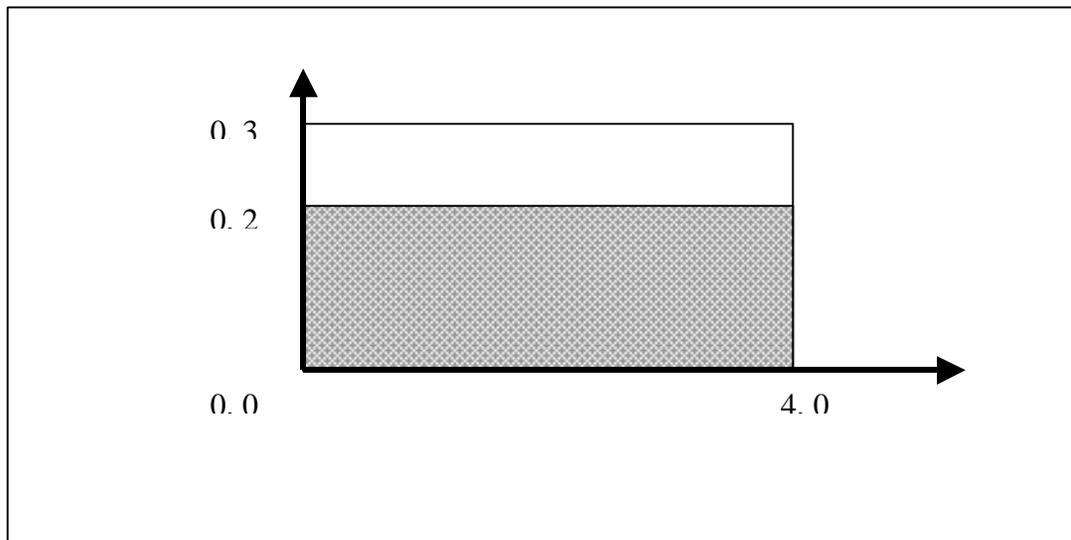
Name _____

In Computer Science

Lab 1

Date _____

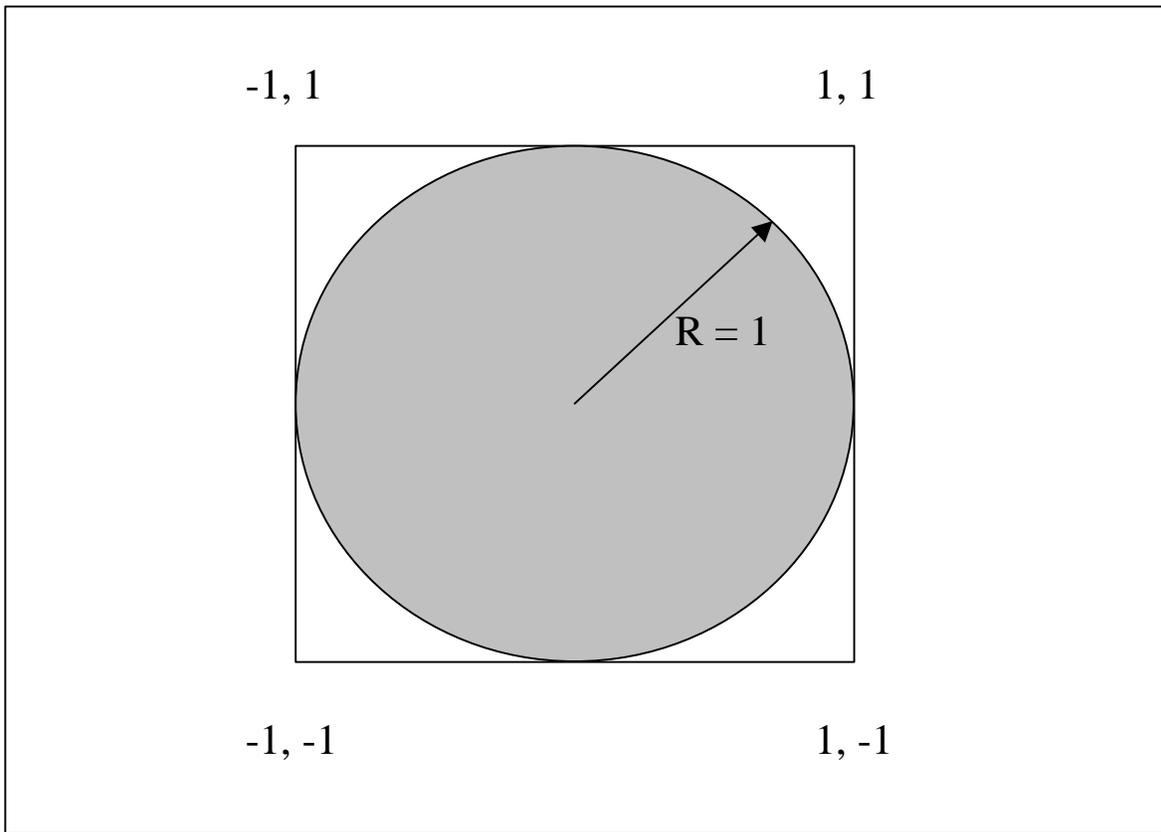
1. Given the following rectangles, modify the program, Shell 1, to simulate the throwing of n darts at the large rectangle. Each dart is equally likely to hit any spot within the large rectangular region.



2. Run your program with different values of n , the number of darts thrown. Complete the following table.

Number of Darts (n)	Number of Darts in the Shaded Region	Ratio	Estimated Area
1,000			
5,000			
10,000			
15,000			
20,000			

3. Given the following unit circle, modify the program, Shell 1, to simulate throwing n darts at the square area. Each dart is equally likely to hit any spot within the square.



4. Run your program with different values of n , the number of darts thrown, to complete the following table.

Number of Darts (n)	Number of Darts in the Shaded Region	Ratio	Estimated Area
1,000			
5,000			
10,000			
15,000			
20,000			

What does the area represent? _____

//Shell 1 - For use with Lab1 - Replace ????? with appropriate code

```
#include <iostream.h>
#include <stdlib.h>

void randomXY(double &x, double &y);
//Pre: x and y are declared
//Post: Random values between 0 and 1 are returned
// for x and y coordinates.
double ratio(double x1, double y1, double x2, double y2, int n);
//Pre: x1, y1 are the lower-right coordinates and x2, y2 are
// the upper-left coordinates of an rectangle enclosing region
// of interest for the function. n is number of random points used.
//Post: Ratio of random points on or below the function
// to total random points in rectangle is returned.

int main()
{
double Ratio, Area, RatioTotal = 0, AreaTotal = 0;
double x1 = ??, y1 =?? ; // lower-left coordinates of rectangle
double x2 = ??, y2 =??; // upper-right coordinates of rectangle
int n = ??; // Number of tries
randomize();
for (int j= 0; j<n; j++) // Loop for multiple tests
{ // of same conditions
Ratio = ratio (x1, y1, x2, y2, 30000);
RatioTotal +=Ratio; // Sum of ratios
Area = ??????????; // Area under the curve
AreaTotal ??????????; // Sum of areas
cout <<"Ratio of areas is: "<< ?????? <<endl
<<" The area within the curve is "<<??????<<endl;
}
Ratio = ??????????????; // average of ratios
Area = ??????????????; // average of areas
cout <<endl<<"For "<<n<<" tries, the average ratio was "<<Ratio
<<" and the Area was "<<Area<<endl;
return 0;
}

void randomXY(double &x, double &y)
{
x = ??????????????;
y = ??????????????;
}

double ratio(double x1, double y1, double x2, double y2, int n)
{double x, y, count=0;
for (int i = 0; i< n; i++)
{ randomXY(x,y);
x = x*(x2 - x1) + x1; // Expand random coordinates to width and length
y = ??????????????; // of rectangle and zero on lower-left
coordinate
?????????????????; // Enter appropriate test here. May be
} // multiple lines.
return count/n;
}
```

Elemental Monte Carlo Methods

Name_____

In Computer Science

Lab 2

Date_____

Prologue: Have you ever stood on a tenth floor balcony and looked down? Or, have you ever stood on a bridge and looked down only to see miles of water? Did you have the urge to drop a penny and watch it fall until it hit the ground or water? Did you wonder how long it would take the penny to hit the ground or water? Or, how fast do you think the penny was traveling? What if you dropped a golf ball and a tennis ball simultaneously, which one would hit the ground or water first? Believe it or not, these everyday questions have simple answers. You have had the knowledge and skills to answer these questions for a very long time. Do you try to answer these questions while you are watching the penny fall? Well, if not, maybe it is time to start!

Did you know that you have something in common with scientists, astronauts, mathematicians, engineers, pilots, truck drivers, track coaches, and ...? They all are concerned with how fast something or someone can travel, the rate of acceleration, distanced traveled, or how long it may take to go from point A to point B. Pilots are concerned with how quickly they can get you from one city to the next, safely of course. Engineers are concerned with how quickly they can get the astronauts to the moon. You and your friends are concerned with how fast you must run the break the school record for the mile or how long it will take you to get to your favorite amusement park.

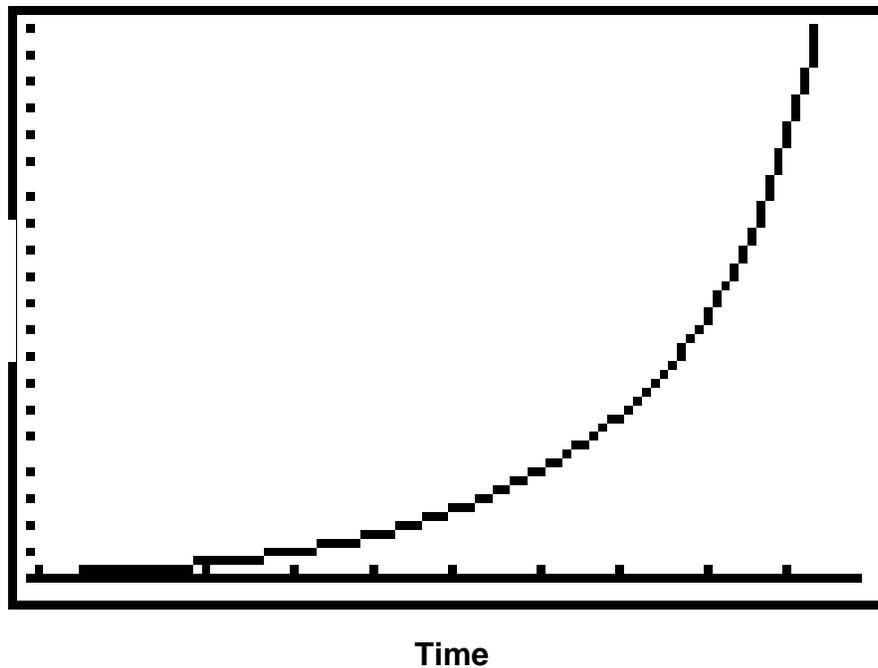
Well, what do you think we can conclude from these examples? Of course, math and science is way of life and we use them everyday!

Problem: Today we are going to examine the flight pattern of an experimental rocket. The rocket will generate 600,000 pounds of thrust. Its initial weight is 500,000 pounds of which 400,000 pounds is fuel. It will consume fuel at a rate of 120,000 pounds per minute.

The following equation represents the velocity of the rocket:

$$Velocity = \left(\frac{thrust}{weight - fuelflow * time} - 1 \right) * gravity * time$$

The graphical representation of the rocket's velocity would look like this:



• Complete:

1. How long into the flight before the engine will shut down?
2. How high will the rocket be at shut down? Write an algorithm and program to estimate this.

**Elemental Monte Carlo Methods
In Computer Science
Lab 3**

Name_____

Date_____

1. Using a Monte Carlo simulation, estimate the probability that at least two people in a random group of people have birthdays on the same day. For simplicity, disregard leap years.

Suggestions:

- Use an apvector of integers to record whether a birthday occurs on a given date.
- Use the random function to determine the apvector index and increment the apvector element to indicate a birthday. Repeat for the number of members in the group.
- Determine the number of days for which more than one person has a birthday.
- Repeat procedure multiple times to obtain the probability.

(Note. You might chose to use the apvector to represent the group and the random function to generate the birthday for each element. Then you would check for members (elements) having the same birthday.)

2. Run your program several times to complete the table below.

Members in Group	Probability of same Birthday
20	
30	
40	

Lab Solutions

//Lab 1 - Part 1 - Find Area of Shaded Rectangle

```
#include <iostream.h>
#include <stdlib.h>

void randomXY(double &x, double &y);
//Pre:   x and y are declared
//Post:  Random values between 0 and 1 are returned
//       for x and y coordinates.

double ratio(double x1, double y1, double x2, double y2, int n);
//Pre:   x1, y1 are the lower-right coordinates and x2, y2 are
//       the upper-left coordinates of an rectangle enclosing region
//       of interest for the function.  n is number of random points used.
//Post:  Ratio of random points on or below the function
//       to total random points in rectangle is returned.

int main()
{
double Ratio, Area, RatioTotal = 0, AreaTotal = 0;
double x1 = 0, y1 = 0;    // lower-left coordinates of rectangle
double x2 = 3, y2 = 4;    // upper-right coordinates of rectangle
int n = 10;                // Number of tries
randomize();

for (int j= 0; j<n; j++)    // Loop for multiple tests
{
Ratio = ratio (x1, y1, x2, y2, 10000); // of same conditions
RatioTotal +=Ratio;
Area = Ratio * (x2 - x1) * (y2 - y1);
AreaTotal +=Area;
cout <<" Ratio of areas is: " << Ratio <<endl
    <<"The area under the curve is " << Area<<endl;
}
Ratio = RatioTotal/n;
Area = AreaTotal/n;
cout <<endl<<"For " <<n<<" tries, the average ratio was " <<Ratio
    <<" and the Area was " <<Area<<endl;
return 0;
}

void randomXY(double &x, double &y)
{
x = random(32767)/32767.0;
y = random(32767)/32767.0;
}
```

```

double ratio(double x1, double y1, double x2, double y2, int n)
{
double x=0, y=0, count=0;
for (int i = 0; i< n; i++)
    {
        randomXY(x,y);
        x = x*(x2 - x1) + x1; // Expand random coordinates to width and
length
        y = y*(y2 - y1) + y1; // of rectangle and zero on lower-left
coordinate.
        if ( y <= 3) // Test for shaded rectangular area
            count++;
    }
return count/n;
}

```

/* Output

```

    Ratio of areas is: 0.7458
The area under the curve is 8.9496
    Ratio of areas is: 0.7507
The area under the curve is 9.0084
    Ratio of areas is: 0.751
The area under the curve is 9.012
    Ratio of areas is: 0.7542
The area under the curve is 9.0504
    Ratio of areas is: 0.7534
The area under the curve is 9.0408
    Ratio of areas is: 0.7526
The area under the curve is 9.0312
    Ratio of areas is: 0.7466
The area under the curve is 8.9592
    Ratio of areas is: 0.7485
The area under the curve is 8.982
    Ratio of areas is: 0.7574
The area under the curve is 9.0888
    Ratio of areas is: 0.7466
The area under the curve is 8.9592

For 10 tries, the average ratio was 0.75068 and the Area was 9.00816

*/

```

Lab Solutions

//Lab1 - Part 2 - Find the Area of the Unit Circle

```
#include <iostream.h>
#include <stdlib.h>

void randomXY(double &x, double &y);
//Pre:   x and y are declared
//Post:  Random values between 0 and 1 are returned
//       for x and y coordinates.

double ratio(double x1, double y1, double x2, double y2, int n);
//Pre:   x1, y1 are the lower-right coordinates and x2, y2 are
//       the upper-left coordinates of an rectangle enclosing region
//       of interest for the function.  n is number of random points used.
//Post:  Ratio of random points on or below the function
//       to total random points in rectangle is returned.

int main()
{
double Ratio, Area, RatioTotal = 0, AreaTotal = 0;
double x1 = -1, y1 = -1;    // lower-left coordinates of rectangle
double x2 = 1, y2 = 1;     // upper-right coordinates of rectangle
int n = 10;                 // Number of tries
randomize();

for (int j= 0; j<n; j++)    // Loop for multiple tests
{
Ratio = ratio (x1, y1, x2, y2, 30000);
RatioTotal +=Ratio;
Area = Ratio * (x2 - x1) * (y2 - y1);
AreaTotal +=Area;
cout <<" Ratio of areas is: " << Ratio <<endl
    <<"The area within the curve is " << Area<<endl;
}
Ratio = RatioTotal/n;
Area = AreaTotal/n;
cout <<endl<<"For " <<n<<" tries, the average ratio was " <<Ratio
    <<" and the Area was " <<Area<<endl;
return 0;
}

void randomXY(double &x, double &y)

{
x = random(32767)/32767.0;
y = random(32767)/32767.0;
}
```

```

double ratio(double x1, double y1, double x2, double y2, int n)
{
double x=0, y=0, count=0;
for (int i = 0; i< n; i++)
{
    randomXY(x,y);
    x = x*(x2 - x1) + x1;    // Expand random coordinates to width and
length
    y = y*(y2 - y1) + y1;    // of rectangle and zero on lower-left
coordinate.
    if ( y * y + x * x -1 <= 0)        // Test for shaded circular area
        count++;
}
return count/n;
}

```

/* Output

```

    Ratio of areas is: 0.786367
The area within the curve is 3.14547
    Ratio of areas is: 0.786433
The area within the curve is 3.14573
    Ratio of areas is: 0.783167
The area within the curve is 3.13267
    Ratio of areas is: 0.783433
The area within the curve is 3.13373
    Ratio of areas is: 0.7842
The area within the curve is 3.1368
    Ratio of areas is: 0.7878
The area within the curve is 3.1512
    Ratio of areas is: 0.7839
The area within the curve is 3.1356
    Ratio of areas is: 0.7836
The area within the curve is 3.1344
    Ratio of areas is: 0.783067
The area within the curve is 3.13227
    Ratio of areas is: 0.781267
The area within the curve is 3.12507

For 10 tries, the average ratio was 0.784323 and the Area was 3.13729

*/

```

Lab Solutions

//Lab 2 Solution - Rocket Problem

```
/* Given the equation for velocity of a rocket as
velocity = thrust/(weight - fuelflow * time) * time - gravity * time.
For this problem thrust = 8,000,000 lbs, weight = 500,000 lbs, and
fuelflow is 120,000 lbs/min or 2,000 lbs/sec and there is 400,000 lbs
of fuel available.
```

```
The time of burnout will be 200 seconds. The corresponding velocity
is 16,000 ft/sec. For convenience, an upper right coordinate of
200, 20000 will be used. (Any value 16,000 or greater could be
used for the y-coordinate.
```

```
*/
```

```
#include <iostream.h>
```

```
#include <stdlib.h>
```

```
void randomXY(double &x, double &y);
```

```
//Pre: x and y are declared
```

```
//Post: Random values between 0 and 1 are returned
```

```
// for x and y coordinates.
```

```
double ratio(double x1, double y1, double x2, double y2, int n);
```

```
//Pre: x1, y1 are the lower-right coordinates and x2, y2 are
```

```
// the upper-left coordinates of an rectangle enclosing region
```

```
// of interest for the function. n is number of random points used.
```

```
//Post: Ratio of random points on or below the function
```

```
// to total random points in rectangle is returned.
```

```
int main()
```

```
{
```

```
double Ratio, Height, RatioTotal = 0, HeightTotal = 0;
```

```
double x1 = 0, y1 = 0; // Starting coordinates
```

```
double x2 = 200, y2 = 20000; // Coordinates at burnout
```

```
int n = 10; // Number of tries
```

```
randomize();
```

```
for (int j= 0; j<n; j++) // Loop for multiple tests
```

```
{ // of same conditions
```

```
Ratio = ratio (x1, y1, x2, y2, 30000);
```

```
RatioTotal +=Ratio;
```

```
Height = Ratio*(x2 - x1)*(y2 - y1)/5280; // Calculate area under curve
```

```
// and convert to miles
```

```
HeightTotal +=Height;
```

```
cout <<" Ratio of areas is: "<< Ratio <<endl
```

```
<<"The height at burnout is "<< Height<<endl;
```

```
}
```

```
Ratio = RatioTotal/n;
```

```
Height = HeightTotal/n;
```

```
cout <<endl<<"For "<<n<<" tries, the average ratio was "<<Ratio
```

```
<<" and the height was "<<Height<<endl;
```

```
return 0;
```

```
}
```

```

void randomXY(double &x, double &y)
{
x = random(32767)/32767.0;          //Convert integer output to double
y = random(32767)/32767.0;
}

double ratio(double x1, double y1, double x2, double y2, int n)
{
double x=0, y=0, count=0;
for (int i = 0; i< n; i++)
{
    randomXY(x,y);
    x = x*(x2 - x1) + x1;    // Expand random coordinates to width and
length
    y = y*(y2 - y1) + y1;    // of rectangle and zero on lower-left
coordinate.
    if (x * 16* (600000/(500000-2000*x) -1) - y >= 0) //Velocity equation
        count ++;
}
return count/n;
}

/* Output

Ratio of areas is: 0.1613
The height at burnout is 122.197
Ratio of areas is: 0.1655
The height at burnout is 125.379
Ratio of areas is: 0.159167
The height at burnout is 120.581
Ratio of areas is: 0.159067
The height at burnout is 120.505
Ratio of areas is: 0.1638
The height at burnout is 124.091
Ratio of areas is: 0.163033
The height at burnout is 123.51
Ratio of areas is: 0.167333
The height at burnout is 126.768
Ratio of areas is: 0.163167
The height at burnout is 123.611
Ratio of areas is: 0.1657
The height at burnout is 125.53
Ratio of areas is: 0.1631
The height at burnout is 123.561

For 10 tries, the average ratio was 0.163117 and the height was 123.573

*/

```

Lab Solutions

/* Lab 3 - Probability of Same Birthday

```
This lab finds the probability of two people in a group having the
same birthday.
*/

#include <iostream.h>
#include <stdlib.h>
#include <apvector.h>

void addBirthdays (apvector <int> &year, int num);
//Pre:   year is declared with 365 elements.  num is defined as the
//       number of people in the group.
//Post:  elements of year will be randomly incremented to simulate
//       a birthday of a given date.

int searchDates (const apvector <int> &year);
//Pre:   year is defined with elements indicating how many birthdays
//       occurs on a given date.
//Post:  Function returns 1 if a birthday occurs more than once.

int main ()
{
    int number;                // size of group
    randomize();              // Generate random seed for random
function

    while(1)                  // Big loop for multiple runs
    {
        double totalMatches = 0;    // number of birthday matches
        cout << " \n\nWhat is the size of the group (-1 to quit): ";
        cin >> number;
        if (number == -1)
            break;
        for (int n = 0; n < 1000; n++)
        {
            apvector <int> dates(365, 0);    // date declared and initialized
at 0
            addBirthdays (dates, number);
            totalMatches += searchDates (dates);
        }
        cout << "The probability of matching birthdays is: "
             << (totalMatches)/1000<<endl;
    }
    return 0;
}

void addBirthdays (apvector <int> &year, int num)
{
    for(int i = 0; i < num; i++)
        year[random(365)]++;
}
```

```
int searchDates (const apvector <int> &year)
{
    for (int i = 0; i < 365; i++)
        if (year[i] > 1)
            return 1;
    return 0;
}
```

/* Output

What is the size of the group (-1 to quit): 20
The probability of matching birthdays is: 0.429

What is the size of the group (-1 to quit): 30
The probability of matching birthdays is: 0.701

What is the size of the group (-1 to quit): 40
The probability of matching birthdays is: 0.897

What is the size of the group (-1 to quit): -1
*/

Elemental Monte Carlo Method – Lab 1A Using a Spreadsheet

PROBLEM: To find the area of the shaded rectangle.

TECHNIQUE: Generate Random numbers within a rectangle counting the points that fall within the shaded region. The ratio of points within total points should equal the respective areas.



Method: The formula used for generating the random x-coordinates is =Rand()*4 and the random y-coordinate is =Rand()*3. The output will be tested against the equation of a rectangle to determine if on or within the shaded rectangle. If true, a one will be returned. These are summed and divided by the number of iterations to get the ratio of the areas. The area of the shaded region was the calculated by multiplying the area of the larger region by the ratio.

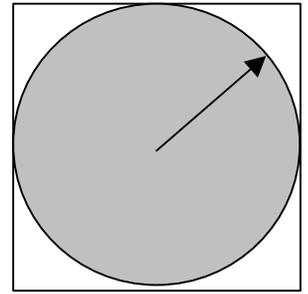
Number of Iterations	Random 'x'	Random 'y'	Test 0 = No 1 = Yes	Total Points with YES (1)	Ratio	Area of the Shaded Region
1	2.342086	2.512774	0	0	.0000	.0000
2	.831967	.589835	1	1	.5000	6.0000
3	3.591924	.402731	1	2	.6667	8.0000
6	1.133593	1.857807	1	3	.5000	6.0000
5	1.896600	.157458	1	4	.8000	9.6000
6	1.859915	2.032566	0	4	.6667	8.0000
7	3.069154	.467588	1	5	.7143	8.5714

Number of Iterations	Random 'x'	Random 'y'	Test 0 = No 1 = Yes	Total Points with YES (1)	Ratio	Area of the Shaded Region
1	=RAND()*4	=RAND()*3	=IF(C19<=2,1,0)	=D19	=E19/A19	=12*F19
2	=RAND()*4	=RAND()*3	=IF(C20<=2,1,0)	=E19+D20	=E20/A20	=12*F20

Elemental Monte Carlo method – Lab 1B Using a Spreadsheet

Problem to find the area in a unit circle using Monte Carlo methods

Technique is to generate random numbers within a square enclosing the unit circle and to count those falling within the circle. The ratio of points within to total points should equal the ratio of the respective areas.



The formula used for generating the random coordinates is $=\text{RAND}()*2 - 1$. The output will be tested against the equation of a circle to determine if on or within the circle. If so, a one will be returned. These are summed and divided by number of iterations to get the ratio of areas.

iterations	random x	random y	test	points within circle	ratio of points	Area of circle
1	0.118813	-0.719368	1	1	1.000000	4.0000
2	-0.493962	-0.774098	1	2	1.000000	4.0000
3	0.007375	0.849107	1	3	1.000000	4.0000
4	-0.493227	-0.707415	1	4	1.000000	4.0000
5	0.444345	0.332874	1	5	1.000000	4.0000
6	-0.421598	-0.091118	1	6	1.000000	4.0000
7	-0.448168	0.961725	0	6	0.857143	3.4286
8	-0.480206	0.321219	1	7	0.875000	3.5000
9	-0.885788	-0.906082	0	7	0.777778	3.1111
10	-0.212889	-0.364623	1	8	0.800000	3.2000
11	-0.544367	0.194043	1	9	0.818182	3.2727
12	0.120948	0.254623	1	10	0.833333	3.3333
13	-0.928332	-0.728447	0	10	0.769231	3.0769
14	0.342411	-0.757340	1	11	0.785714	3.1429
15	0.587820	-0.596833	1	12	0.800000	3.2000
16	0.929915	-0.885199	0	12	0.750000	3.0000
17	0.886786	-0.447817	1	13	0.764706	3.0588
18	0.412987	0.368748	1	14	0.777778	3.1111
19	-0.104506	0.071658	1	15	0.789474	3.1579

iterations	random x	random y	test	points within circle	ratio of points	area of circle
1	$=\text{RAND}()*2 - 1$	$=\text{RAND}()*2 - 1$	$=\text{IF}(B15*B15+C15*C15-1<=0,1,0)$	$=D15$	$=E15/A15$	$=4*F15$
2	$=\text{RAND}()*2 - 1$	$=\text{RAND}()*2 - 1$	$=\text{IF}(B16*B16+C16*C16-1<=0,1,0)$	$=E15+D16$	$=E16/A16$	$=4*F16$
3	$=\text{RAND}()*2 - 1$	$=\text{RAND}()*2 - 1$	$=\text{IF}(B17*B17+C17*C17-1<=0,1,0)$	$=E16+D17$	$=E17/A17$	$=4*F17$

Elemental Monte Carlo Methods in Computer Science

Using Spread sheets

Lab 1A

PROBLEM: To find the area of the enclosed rectangle.

TECHNIQUE: Generate Random numbers within a rectangle counting the points that fall within the shaded region. The ratio of points within to total points should equal the respective areas.

Method: The formula used for generating the random x-coordinates is $=\text{Rand}()*4$ and the random y-coordinate is $=\text{Rand}()*3$. The output will be tested against the equation of a rectangle to determine if on or within the shaded rectangle. If true, a one will be returned. These are summed and divided by the number of iterations to get the ratio of the areas. The area of the shaded region was the calculated by multiplying the area of the larger region by the ratio.

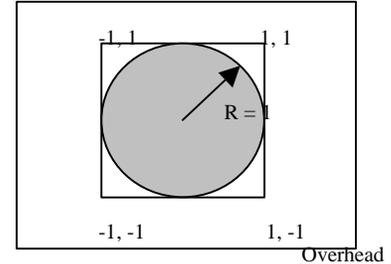
Number of Iterations	Random 'x'	Random 'y'	Test 0 = No 1 = Yes	Total Points with YES (1)	Ratio
1	3.229947	2.860649	0	0	.0000
2	3.348478	.071826	1	1	.5000
3	2.248865	1.036554	1	2	.6667
6	2.950945	1.632011	1	3	.5000
5	2.072826	2.617139	0	3	.6000
6	2.065548	2.555150	0	3	.5000
7	2.590283	2.761697	0	3	.4286
8	.010726	.257531	1	4	.5000
9	2.182977	.843839	1	5	.5556
10	3.864213	1.977154	1	6	.6000
11	3.281221	2.268200	0	6	.5455
12	1.714302	1.621786	1	7	.5833
13	1.008820	1.198903	1	8	.6154
14	.463819	.057039	1	9	.6429
15	.010647	2.815088	0	9	.6000
16	1.412125	2.615536	0	9	.5625
17	1.327121	1.811326	1	10	.5882
18	2.499661	2.662255	0	10	.5556
19	2.012328	1.635576	1	11	.5789
20	3.923144	1.629803	1	12	.6000
21	2.040518	1.927566	1	13	.6190
22	.770815	1.263228	1	14	.6364
23	3.023650	1.791704	1	15	.6522
24	.111002	1.957448	1	16	.6667
25	3.000413	.730751	1	17	.6800
26	3.901323	1.341401	1	18	.6923
27	3.734501	1.534766	1	19	.7037
28	1.376068	1.570494	1	20	.7143
29	3.209838	1.460439	1	21	.7241
30	.664603	.603682	1	22	.7333

31	3.906483	1.395200	1	23	.7419
32	2.416463	1.896051	1	24	.7500
33	.932606	.747614	1	25	.7576
34	3.972070	1.947193	1	26	.7647
35	2.542501	2.016496	0	26	.7429
36	2.747459	1.636826	1	27	.7500
37	3.644492	1.905796	1	28	.7568
38	1.404642	.461013	1	29	.7632
39	3.837838	1.245175	1	30	.7692
40	.249612	1.499534	1	31	.7750
41	.288830	.555257	1	32	.7805
42	1.443983	2.810365	0	32	.7619
43	2.204322	1.224178	1	33	.7674
44	1.491061	2.290961	0	33	.7500
45	.558066	.160345	1	34	.7556
46	1.613584	.384805	1	35	.7609
47	3.849129	2.802003	0	35	.7447
48	3.137963	.105786	1	36	.7500
49	2.784816	1.562126	1	37	.7551
50	2.521936	.272836	1	38	.7600
51	.774790	1.152623	1	39	.7647
52	2.053008	1.542475	1	40	.7692
53	3.439996	2.285734	0	40	.7547
54	1.050902	2.831350	0	40	.7407
55	1.753413	.317952	1	41	.7455
56	1.117989	.236416	1	42	.7500
57	.617550	.474370	1	43	.7544
58	1.806389	.423655	1	44	.7586
59	1.395017	.346872	1	45	.7627
60	1.310318	2.261182	0	45	.7500
61	1.477213	1.700385	1	46	.7541
62	.550307	2.111870	0	46	.7419
63	1.410302	2.307718	0	46	.7302
64	3.420340	2.734701	0	46	.7188
65	.006248	2.136445	0	46	.7077
66	2.207280	.655586	1	47	.7121
67	2.537326	1.142749	1	48	.7164
68	3.920259	.571872	1	49	.7206
69	1.704373	.990120	1	50	.7246
70	3.493866	1.597658	1	51	.7286
71	2.105739	1.231078	1	52	.7324
72	1.303190	1.050090	1	53	.7361
73	3.471245	2.563266	0	53	.7260
74	3.937557	1.471931	1	54	.7297
75	3.588419	2.472959	0	54	.7200
76	2.711416	2.567689	0	54	.7105
77	1.663270	2.905238	0	54	.7013
78	3.817444	2.976462	0	54	.6923
79	3.959315	.462513	1	55	.6962
80	1.281373	.971401	1	56	.7000
81	1.209588	.762788	1	57	.7037

82	213903	.979431	1	58	.7073
83	1.590552	1.627656	1	59	.7108
84	1.660292	1.841093	1	60	.7143
85	2.572727	1.013088	1	61	.7176
86	3.594381	2.172696	0	61	.7093
87	1.259599	2.678917	0	61	.7011
88	3.366762	.275279	1	62	.7045
89	1.819034	1.153126	1	63	.7079
90	1.577250	.479920	1	64	.7111
91	3.019645	.926666	1	65	.7143
92	.780578	2.425029	0	65	.7065
93	2.900009	2.840812	0	65	.6989
94	1.832320	.250902	1	66	.7021
95	.063786	1.746590	1	67	.7053
96	3.290760	2.891282	0	67	.6979
97	3.402475	.910936	1	68	.7010
98	.552572	.936176	1	69	.7041
99	3.650507	2.819044	0	69	.6970
100	.473107	2.466539	0	69	.6900

Problem to find the area in a unit circle using Monte Carlo methods:

Technique is to generate random numbers within a square enclosing the unit circle and to count those falling within the circle. The ratio of points within to total points should equal the ratio of the respective areas.



The formula used for generating the random coordinates is $=\text{RAND}()*2 - 1$. The output will be tested against the equation of a circle to determine if on or within the circle. If so, a one will be returned. These are summed and divided by number of iterations to get the ratio of areas.

Iterations	Random x	Random y	Test	Points within Circle	Ratio of Points	Area of Circle
1	0.119824	-0.783839	1	1	1.000000	4.0000
2	0.336329	-0.487052	1	2	1.000000	4.0000
3	-0.910826	-0.800526	0	2	0.666667	2.6667
4	0.455120	0.637132	1	3	0.750000	3.0000
5	0.157420	0.446419	1	4	0.800000	3.2000
6	0.701780	0.608744	1	5	0.833333	3.3333
7	0.901156	0.674781	0	5	0.714286	2.8571
8	-0.549261	0.126575	1	6	0.750000	3.0000
9	-0.486138	0.057378	1	7	0.777778	3.1111
10	-0.068013	0.468676	1	8	0.800000	3.2000
11	-0.709646	0.993599	0	8	0.727273	2.9091
12	0.556955	0.275696	1	9	0.750000	3.0000
13	0.036719	-0.963359	1	10	0.769231	3.0769
14	-0.721910	0.539851	1	11	0.785714	3.1429
15	0.300079	-0.505139	1	12	0.800000	3.2000
16	-0.543143	0.214887	1	13	0.812500	3.2500
17	0.824751	0.301706	1	14	0.823529	3.2941
18	-0.523303	-0.938587	0	14	0.777778	3.1111
19	0.554960	0.681223	1	15	0.789474	3.1579
20	0.718037	0.268579	1	16	0.800000	3.2000
21	0.134894	-0.781490	1	17	0.809524	3.2381
22	-0.592339	-0.943375	0	17	0.772727	3.0909
23	-0.466726	0.711284	1	18	0.782609	3.1304
24	-0.057093	-0.284109	1	19	0.791667	3.1667
25	0.185993	-0.936294	1	20	0.800000	3.2000
26	-0.921430	-0.940360	0	20	0.769231	3.0769
27	-0.855011	-0.638633	0	20	0.740741	2.9630
28	0.411294	-0.262082	1	21	0.750000	3.0000
29	0.520029	-0.368176	1	22	0.758621	3.0345
30	0.564855	-0.134339	1	23	0.766667	3.0667
31	-0.923989	-0.068810	1	24	0.774194	3.0968
32	0.640371	-0.494969	1	25	0.781250	3.1250
33	-0.668327	0.788060	0	25	0.757576	3.0303
34	-0.042826	-0.175656	1	26	0.764706	3.0588
35	-0.692494	-0.557965	1	27	0.771429	3.0857

36	0.649952	-0.399398	1	28	0.777778	3.1111
37	-0.061788	-0.398773	1	29	0.783784	3.1351
38	0.075072	-0.294290	1	30	0.789474	3.1579
39	0.203913	-0.947486	1	31	0.794872	3.1795
40	-0.840426	0.596969	0	31	0.775000	3.1000
41	-0.745744	0.471524	1	32	0.780488	3.1220
42	-0.735760	0.520099	1	33	0.785714	3.1429
43	0.313275	-0.899309	1	34	0.790698	3.1628
44	0.308000	-0.713041	1	35	0.795455	3.1818
45	-0.350724	-0.038741	1	36	0.800000	3.2000
46	-0.047814	0.837515	1	37	0.804348	3.2174
47	-0.343005	-0.225889	1	38	0.808511	3.2340
48	-0.037605	-0.894163	1	39	0.812500	3.2500
49	0.851008	0.173639	1	40	0.816327	3.2653
50	-0.269980	0.945028	1	41	0.820000	3.2800
51	-0.453619	-0.564914	1	42	0.823529	3.2941
52	0.399570	0.596346	1	43	0.826923	3.3077
53	-0.858962	0.560507	0	43	0.811321	3.2453
54	-0.841887	0.247014	1	44	0.814815	3.2593
55	0.350078	0.540786	1	45	0.818182	3.2727
56	-0.520649	-0.873468	0	45	0.803571	3.2143
57	0.090408	0.315851	1	46	0.807018	3.2281
58	0.390743	-0.095243	1	47	0.810345	3.2414
59	-0.954918	0.171231	1	48	0.813559	3.2542
60	0.082762	-0.775320	1	49	0.816667	3.2667
61	0.009124	0.029930	1	50	0.819672	3.2787
62	0.361483	0.546451	1	51	0.822581	3.2903
63	-0.877635	-0.827667	0	51	0.809524	3.2381
64	-0.097082	0.979767	1	52	0.812500	3.2500
65	0.716546	-0.378679	1	53	0.815385	3.2615
66	-0.588548	0.288752	1	54	0.818182	3.2727
67	0.251806	-0.594990	1	55	0.820896	3.2836
68	-0.971240	-0.123046	1	56	0.823529	3.2941
69	-0.009221	-0.138467	1	57	0.826087	3.3043
70	0.537813	0.288406	1	58	0.828571	3.3143
71	0.855141	0.759530	0	58	0.816901	3.2676
72	-0.233482	-0.601515	1	59	0.819444	3.2778
73	0.942629	-0.018242	1	60	0.821918	3.2877
74	-0.732521	0.330265	1	61	0.824324	3.2973
75	-0.041220	-0.395561	1	62	0.826667	3.3067
76	-0.386048	-0.954698	0	62	0.815789	3.2632
77	0.338314	-0.991221	0	62	0.805195	3.2208
78	-0.360683	0.158748	1	63	0.807692	3.2308
79	-0.509925	0.446784	1	64	0.810127	3.2405
80	0.292731	-0.665566	1	65	0.812500	3.2500
81	-0.105638	0.948130	1	66	0.814815	3.2593
82	0.004792	-0.519003	1	67	0.817073	3.2683
83	-0.707901	0.131157	1	68	0.819277	3.2771
84	0.511064	-0.416483	1	69	0.821429	3.2857
85	0.145569	0.053116	1	70	0.823529	3.2941
86	0.073969	0.870706	1	71	0.825581	3.3023
87	-0.377713	0.900416	1	72	0.827586	3.3103

88	-0.594176	-0.982067	0	72	0.818182	3.2727
89	-0.459154	-0.927781	0	72	0.808989	3.2360
90	0.688398	-0.817474	0	72	0.800000	3.2000
91	0.011324	-0.359627	1	73	0.802198	3.2088
92	0.523419	0.918952	0	73	0.793478	3.1739
93	-0.553402	-0.535333	1	74	0.795699	3.1828
94	0.920525	0.897028	0	74	0.787234	3.1489
95	0.130588	0.930926	1	75	0.789474	3.1579
96	-0.955661	0.874629	0	75	0.781250	3.1250
97	0.150047	0.035230	1	76	0.783505	3.1340
98	0.421241	-0.565682	1	77	0.785714	3.1429
99	0.862935	-0.687921	0	77	0.777778	3.1111
100	0.353702	0.132316	1	78	0.780000	3.1200
101	0.860308	-0.490612	1	79	0.782178	3.1287
102	-0.105964	-0.247403	1	80	0.784314	3.1373
103	0.674443	0.128252	1	81	0.786408	3.1456
104	-0.014942	-0.324433	1	82	0.788462	3.1538
105	0.167429	0.741283	1	83	0.790476	3.1619
106	0.566468	-0.296821	1	84	0.792453	3.1698
107	-0.658894	-0.575231	1	85	0.794393	3.1776
108	-0.920876	0.503219	0	85	0.787037	3.1481
109	0.541111	0.675696	1	86	0.788991	3.1560
110	0.433890	-0.348372	1	87	0.790909	3.1636
111	-0.937011	0.038423	1	88	0.792793	3.1712
112	-0.220557	0.329006	1	89	0.794643	3.1786
113	-0.408442	-0.890674	1	90	0.796460	3.1858
114	-0.885128	-0.419235	1	91	0.798246	3.1930
115	-0.883607	0.522134	0	91	0.791304	3.1652
116	0.301879	-0.821710	1	92	0.793103	3.1724
117	0.408591	-0.805590	1	93	0.794872	3.1795
118	0.722174	0.890036	0	93	0.788136	3.1525
119	-0.530240	0.931617	0	93	0.781513	3.1261
120	-0.161942	-0.005646	1	94	0.783333	3.1333
121	0.973909	-0.816765	0	94	0.776860	3.1074
122	0.974209	0.126736	1	95	0.778689	3.1148
123	-0.901133	0.394453	1	96	0.780488	3.1220
124	0.350097	0.723227	1	97	0.782258	3.1290
125	-0.761724	-0.466153	1	98	0.784000	3.1360
126	0.334884	-0.686348	1	99	0.785714	3.1429
127	-0.196280	0.758737	1	100	0.787402	3.1496
128	-0.018803	-0.240882	1	101	0.789063	3.1563
129	0.715900	-0.721033	0	101	0.782946	3.1318
130	-0.846936	0.665466	0	101	0.776923	3.1077
131	-0.034213	0.416973	1	102	0.778626	3.1145
132	-0.483660	0.393762	1	103	0.780303	3.1212
133	-0.440671	0.590257	1	104	0.781955	3.1278
134	-0.662515	-0.137985	1	105	0.783582	3.1343
135	-0.732724	0.340566	1	106	0.785185	3.1407
136	-0.880840	-0.308861	1	107	0.786765	3.1471
137	-0.896449	0.395386	1	108	0.788321	3.1533
138	-0.487167	-0.041766	1	109	0.789855	3.1594
139	0.241807	-0.787262	1	110	0.791367	3.1655