EECS 10: Assignment 5

Prepared by: Ziggy Fan, Prof. Rainer Doemer

October 22, 2010

Due Monday 11/1/2010 12:00pm

1 Roulette [15 points]

Write a program to simulate the casino game Roulette. We will use a simplified version of roulette. We will only use numbers from 0 to 36 (00 is excluded). Further, the user is only allowed to bet on odd or even numbers.
At the start, your program will prompt the user to enter the money she/he has (*balance*) and the *betting type*: *1-Even number, 2-Odd number, 0 to quit the game*. Then your program will prompt for the *betting amount.*

Following this, your program will generate a random number between 0 and 36 (including 0 and 36). Depending on the *betting type* chosen by the user, the program will check the generated number for *even or odd number*. If won, the user gets the *betting amount* which will be added to the *balance*. If lost, the user will lose the *betting amount* which will be deducted from the *balance.*

As long as the user has some *balance*, the program will repeat prompting the user for the next betting. Your program will quit if the user loses all the money (i.e. *balance* is 0), or when the user enters 0 when prompted for the *betting type*. When run, your program should look as follows:

```
Entering the casino, how much money do you have? $100
We are playing Roulette, odd or even bets only.
Place your bet!
Enter 1 for odd, 2 for even, 0 to quit: 1
How much money you want to bet? $20
You bet $20.00 on odd numbers.
The winning number is 23!
You Win!
Your balance is $120.00!!
Place your bet!
Enter 1 for odd, 2 for even, 0 to quit: 2
How much money you want to bet? $100
You bet $100.00 on even numbers.
The winning number is 29!
You Lose!
Your balance is $20.00!!
Place your bet!
Enter 1 for odd, 2 for even, 0 to quit: 0
You exit the casino with $20.00
```

What to turn in:

You should submit your program code as file **`roulette.c`**, a text file **`roulette.txt`** briefly explaining how you designed your program, and a typescript **`roulette.script`** which shows that you compile your program and run it for 5 bets.

HINT

For generating the random number, you have to use a random number generator which is provided by the C standard function **`rand()`**. This function generates a random number of type **`int`** in the range of **`0`** to **`32767`**. This function is provided in the header file **`stdlib.h`**.

In practice, no computer functions can produce truly random data -- they only produce pseudo-random numbers. These are computed from a formula and the number sequences they produce are repeatable. A seed value is usually used by the random number generator to generate a number. Therefore, if you use the same seed value all the time, the same sequence of "random" numbers will be generated (i.e. your program will always produce the same "random" number in every program run). To avoid this, we can use the current time of the day to set the random seed, as this will always be changing with every program run. With this trick, your program will produce different guesses every time you run it.

To set the seed value, you have to use the function **`srand()`**, which is also provided by header file **`stdlib.h`** . For the current time of the day, you can use the function **`time()`**, which is defined in header file **`time.h`** (**`stdlib.h`** and **`time.h`** are header files just like the **`stdio.h`** file that we have been using so far).

In summary, use the following code fragments to generate the random number for the game:

1. Include the **`stdlib.h`** and **`time.h`** header files at the beginning of your program:

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

2. Include the following lines at the beginning of your main function:

```
/* initialize the random number generator with the current time */
srand(time(0));
```

3. In your program, use the following to generate the random number:

```
/* generate the random number in the range 0 to 36 */
randomNumber = rand() % 37;
```

Here, **`randomNumber`** is the integer variable which is assigned the generated random number.

2 Monte Carlo Calculation of Pi [25 points]

Monte Carlo (MC) methods are stochastic techniques, meaning they are based on the use of random numbers and probability statistics to investigate problems. In this part of the homework, you are asked to write a program to implement a simple geometric MC experiment which calculates the value of Pi based on a "hit and miss" integration.

The figure below shows two equally sized semi-circles circumscribed by a square. The radius of the circle *r* equals to 1/2 of the side of the square. Furthermore, the center of each semi-circle is located at a distance *r* above the x-axis. (The center of the semi-circle on the left is located at (0, *r*) and the center of the semi-circle on the right is located at (2*r*, *r*).)

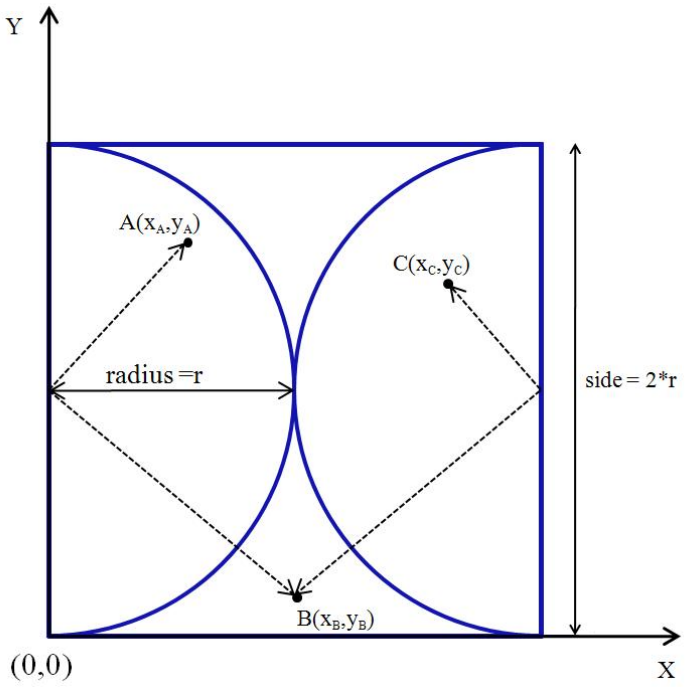

Figure 1: Two Semi-Circles Circumscribed by a Square

Imagine we can throw points randomly at the above figure. If we can throw infinite random points, it should be apparent that of the total number of points that hit the semi-circles divided by the the number of points that hit within the square is proportional to the area of that part.
In other words:

$$\frac{number\ of\ points\ hitting\ area\ of\ semicircles}{number\ of\ points\ hitting\ square\ area} = \frac{area\ of\ 2\ semicircles}{area\ of\ square} = \frac{area\ of\ 1\ circle}{area\ of\ square} = \frac{\pi \times r \times r}{(2 \times r)^2} = \frac{\pi \times r \times r}{4 \times r \times r} = \frac{\pi}{4}$$

Therefore, we get the formula to calculate π using Monte Carlo method:

$$\pi = 4 \times \frac{number\ of\ points\ hitting\ area\ of\ semicircles}{number\ of\ points\ hitting\ square\ area}$$

In the real world, we can only throw a finite number of random points, therefore, the π calculated using the above formula is an approximation of the exact value of π.

We can have our computer generate random numbers to simulate the throwing of points. For each point, we can have the computer generate two random floating point numbers to be the x and y coordinates of the point, where $0 \leq x \leq 2r$ and $0 \leq y \leq 2r$ so that (x,y) must fall within the square area. However, this randomly generated point could fall within one of the two semi-circle areas or fall out of both semi-circle areas.

To decide if the randomly generated point (x,y) is within ONE of the two semi-circle areas, we can compare the distance of the point to the center of both semi-circles with the radius r. For example, the point A in the above figure is in the left semi-circle area since its distance to the center of the left semi-circle is less than r. Likewise, point C (although is not in the left semi-circle), is within the right semi-circles because its distance to the center of the right semi-circle is less than r. However, the point B in the above figure is not in either semi-circle since its distance to the center of both semi-circles is greater than r.

Note: If the distance of point to the center of either semi-circle equals to radius r, then that point is considered within that semi-circle area.

Assume the radius of both semi-circle is r and the coordinates of the randomly generated point P is (x,y), then the distance of P to the center of the left semi-circle for instance is:
$Distance(P,CenterOfLeftSemiCircle) = \sqrt{(x-0)\times(x-0)+(y-r)\times(y-r)}$

To avoid the square root calculation, you can compare $Distance(P,CenterOfLeftSemiCircle)\times Distance(P,CenterOfLeftSemiCircle)$ with the radius squared $r\times r$ in order to decide if the randomly generated point is within the semi-circle area.

At the beginning, your program should ask for the input of radius r and the number of random points N the computer needs to generate. The output should like this:

```
Enter the radius of circles:  5
Enter the number of points:  10
```

During the generation, whenever a random point is generated, your program should print out the coordinates of the point and whether the point is *In* or *Out* the circle area like this:

```
Point No.1(x=0.000000,y=6.551714):  IN
Point No.2(x=3.048189,y=6.749779):  IN
Point No.3(x=1.067537,y=5.165868):  IN
Point No.4(x=4.896695,y=6.024659):  OUT
Point No.5(x=3.699454,y=2.566607):  IN
Point No.6(x=3.741874,y=8.255867):  IN
Point No.7(x=1.727042,y=2.977996):  IN
Point No.8(x=6.435438,y=7.896664):  IN
Point No.9(x=9.878231,y=8.005921):  IN
Point No.10(x=4.642476,y=5.389874):  IN
```

At the end, your program should output the number of points within and out of the circle, together with the approximation value of π like this:

```
/******In Summary******/
Points within circle areas:  9
Points out of circle areas:  1
Pi= 3.600000
```

To show that your program works correctly, run it once with the radius = 5 and the number of points = 35. Submit the output as your script file (**mc.script**). Please compile your C code using **-ansi -Wall** options as below to specify ANSI code with all warnings:

```
gcc -o mc -ansi -Wall mc.c
```

The files that you should submit for this part of the assignment are:

- **mc.c**: the source code file.
- **mc.txt**: the brief text file to explain what the program does and why you chose your method of implementation.
- **mc.script**: the typescript file to show that your program works with the radius = 5 and the number of points = 35.

HINT In assignment 4, you have learned how to generate random integer numbers within the range of $[0,n)$ (n is exclusive). However, in this homework, you need to generate floating point numbers with the range of $[0,n]$ (n is inclusive). Therefore, there are some modification of the code described in homework4.

You need to replace the following line in assignment 4
int randomNumber; with

double randomNumber;
randomNumber = rand() % n; with
*randomNumber = ((double)rand())/((double)RAND_MAX)*s;* /* s is the side of the square */

Furthermore, in assignment 5, we want you to use the same seed for the random numbers generation in order to generate the same series of random numbers. Therefore, take out the following line:
#include ⟨time.h⟩

and replace the following line in homework4
srand(time(NULL)); with
srand(0);

3 Bonus Problem [5 points]

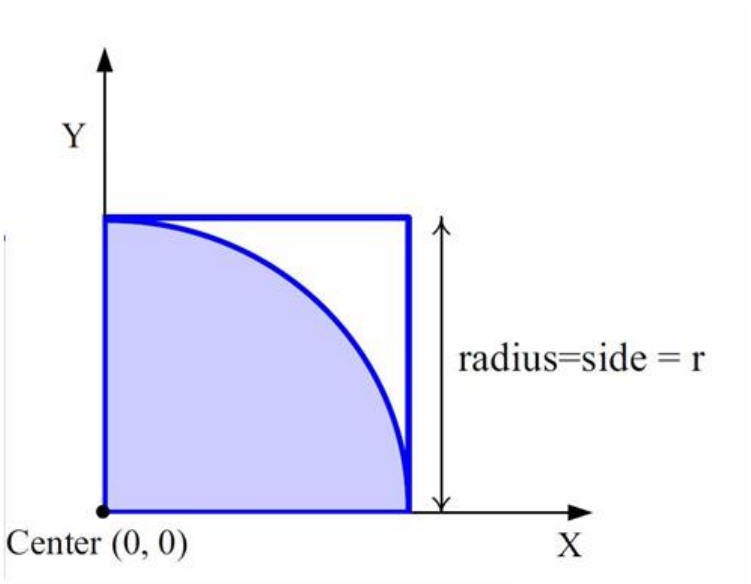

Figure 2: A Quadrant of a Circle Circumscribed by a Square

Could we use the Monte Carlo method to calculate Pi given Figure 2 above? It's a quadrant of a circle circumscribed by a square. The radius of circle equals to the side of square and their centers are overlapped.

If yes, explain your method and formula in the same text file of part 2 (**mc.txt**) concisely within 10 sentences.

4 Submission

Submission for these files will be similar to last week's assignment. The only difference is that you need to create a directory called **`hw5/`**. Put all the files for assignment 5 in that directory and run the **`/ecelib/bin/turnin`** command to submit your homework.

Note: We do require the *exact* file names. If you use different file names, we will not see your files for grading. Also, please pay attention to any announcements on the course noteboard.