CS 171 Computer Programming 1

Winter 2016

Lab 7 – Math and Random Numbers

NAME: Joseph Mulray LAB: 7

Question 1: (3pts)

Some of these early steps don't seem as if they have anything to do with math or random numbers, but we're starting by building some pieces we'll use later.

The first piece you need to **build is a function** that prints a selected vowel. It should take an integer as a parameter. If the parameter is `0`, the function should print `'A'`; if it's `1` print `'E'`, if it's `2` print `'I'`, if it's `3` print `'O'`, and if it's `4` print `'U'`. In CS172, we'll study a way of organizing data called *arrays* which will allow this to be done quite compactly. For now, however, the **switch** statement will be your best approach (or of course you could use an if..else if statement instead).

When you get that function working correctly, paste the function below (we only need to function, though presumably you created some `main()` where you tested it).

Answers:

```
Void printVowel(int vowel)
{
switch (vowel)
{
    case 0:
    cout<<"A";
    break;
    case 1:
    cout<<"E";
    break;
    case 2:
    cout<<"I";
    break;
    case 3:
    cout<<"O";
    break;
    case 4:
    cout<<"U";
    break;
}
}
```

Question 2: (3pts)

Your second task is to **create a function** that will be a tool for randomness in this program. This function should take an integer argument `n` and return a random integer in the range `[0, n-1]`.

Again, when you get this function working and have tested it, paste just that function below.

Answer:

```
int random(int x)
{
   int r;
   do
   {
      r= rand();
   }while( r>x-1 || r<0);
   return r;
}
```

Question 3: (2pts)

Now using the two functions you've just created, have your *main*() (or create one if you haven't already, though you probably should have….) function to print six random vowels. Don't put

any whitespace between them. Also, don't yet do anything to seed the random number generator.

Paste just your `main()` function below.

Answer:

```
int main()
{

   const int num=5;


   for(int x=0;x<6;x++)
   {
     printVowel(random(num));
   }



}
```

Question 4 (1pt)

Run your program twice. Paste below the results of your two runs.

Answer:

UAEAAO

UAEAAO

Question 5: (2pts)

Next, pick your favorite integer. Add a statement to your *main*() function that seeds the random number generator with the number you picked.

Run your program twice. Paste below the results of your two runs.

Answer:

```
AIUIUU
AIUIUU
```

Question 6: (2pts)

For this step, change your seeding of the random number generator to use the time as a seed.

Run your program twice. Paste below the results of your two runs.

Answer:

```
UAEEAO
UOAIOI
```

Question 7: (5pts)

For this part, you are to **create a function** similar to your vowel function for the 21 consonants in the Latin alphabet. You will also need to change your *main*() program so that it prints out random "words" which consist of (without the dashes):

consonant-vowel-consonant-consonant-vowel-consonant

For example:

PEXWIM

Notice that although these aren't dictionary words, they are all pronounceable.

In the answer box below, paste your entire file program.

Answer:

```
// Joseph Mulray
#include <iostream>
#include <string>
#include <cstdlib>
using namespace std;

void printConsonants(int con)
{
    switch(con)
    {
        case 0:
        cout<<"B";
        break;
        case 1:
        cout<<"C";
        break;
        case 2:
```

```
cout<<"D";
break;
case 3:
cout<<"F";
break;
case 4:
cout<<"G";
break;
case 5:
cout<<"H";
break;
case 6:
cout<<"J";
break;
case 7:
cout<<"K";
break;
case 8:
cout<<"L";
break;
case 9:
cout<<"M";
break;
case 10:
cout<<"N";
break;
case 11:
cout<<"P";
break;
case 12:
cout<<"Q";
break;
```

```
        case 13:
        cout<<"R";
        break;
        case 14:
        cout<<"S";
        break;
        case 15:
        cout<<"T";
        break;
        case 16:
        cout<<"V";
        break;
        case 17:
        cout<<"W";
        break;
        case 18:
        cout<<"X";
        break;
        case 19:
        cout<<"Y";
        break;
        case 20:
        cout<<"Z";
        break;
}
}



void printVowel(int vowel)
{
switch (vowel)
```

```
{
    case 0:
    cout<<"A";
    break;
    case 1:
    cout<<"E";
    break;
    case 2:
    cout<<"I";
    break;
    case 3:
    cout<<"O";
    break;
    case 4:
    cout<<"U";
    break;
}
}
int random(int x)
{
    int a;
    a = (rand() % x);
    return a;
}

int main()
{
    int const vow=5;
    int const con=21;
    srand(time(NULL));

    printConsonants(random(con));
```

```
    printVowel(random(vow));
    printConsonants(random(con));
    printConsonants(random(con));
    printVowel(random(vow));
    printConsonants(random(con));

}
```

Question 8: (3pts)

For the remaining parts of this lab we're going to compute π using a technique known as the *Monte Carlo* method. The first component you need is a function that returns a random floating point number in the range of `[-1, 1]`. **Write such a function and a *main*() function to test it.**

When you're satisfied with the results, paste your function and the `main()` below.

Answer:

// Joseph Mulray

```
#include <iostream>
#include <string>
#include <cstdlib>
using namespace std;
float number()
{
   int temp;
   float a;
   temp = (rand() % 2000-1000);
   a = float(temp /1000.0);
   return a;
}

int main()
{
srand(time(NULL));

cout<<number();

return 0;
}
```

Question 9: (3pts)

If we call our random number function twice and get a pair of numbers, we can consider them the Cartesian coordinates `(x,y)` of a point in a square centered on the origin where each side is 2 units long. However, we're going to want to work with that point in polar coordinates (r, θ).

If you're not familiar with polar coordinates, the idea is that the value of `r` gives the distance from the origin (for our purposes, the point `(0,0)`) and θ gives the angle of a line going from the origin to the point `(x,y)`.

For this step, you should **create a function** that transforms rectangular (Cartesian) coordinates to polar coordinates. It should take 4 arguments. The first two should be double precision floating point numbers **passed by value**. You'll pass `x` and `y` into those two positions. The other two parameters should be **reference parameters**, again for double precision floating point numbers. They will be where you'll **put** the values for `r` and .

Before immediately jumping to Google or Wikipedia, discuss in your team and other teams around you how to implement this function. Only resort to those references if you are really stuck.

As usual, implement the function and test it. Then paste just your new function below.

Answer:

```
// Joseph Mulray
#include <iostream>
#include <string>
#include <cstdlib>
#include <cmath>
using namespace std;

float transforms (float x, float y, float &a, float &b)
{
  a=sqrt((x*x) + (y*y));
  b=atan(y/x);

}

float number()
{
   int temp;
   float a;
   temp = (rand() % 2000-1000);
   a = float(temp /1000.0);
   return a;
}

int main()
{
```

```
float x,y,r,t;
srand(time(NULL));

x=number();
y=number();

cout<<x<<endl;
cout<<y<<endl;

transforms(x,y,r,t);
cout<<"Radius: " <<r<<endl;
cout<<"θ: "<<t<<endl;
return 0;
}
```

Question 10: (3pts)

The first use we'll make of the polar coordinates will be to perform a very crude test of how well distributed our random numbers are. In your `main()`, generate 100,000 points, convert them to polar coordinates and print out the ratio of the number of points where $\theta \geq 0$ to the number where $\theta < 0$. If you only allowed for positive angles in the previous part you'll want to check the ratio between when $0 \leq \theta < 180$ and $180 \leq \theta < 360$, or their radian equivalent.

Paste your new `main()` below.

Answer:

```
// Joseph Mulray
#include <iostream>
#include <string>
#include <cstdlib>
#include <cmath>
using namespace std;

void transforms (float x, float y, float &a, float &b)
{
  a=sqrt((x*x) + (y*y));
  b=atan(y/x);

}

float number()
{
   int temp;
   float a;
   temp = (rand() % 2000-1000);
   a = float(temp /1000.0);
```

```
   return a;
}

int main()
{
int positive=0;
int negative=0;
float r,t;
srand(time(NULL));


for(int a=0;a<=100000;a++)
{
  float x=number();
  float y=number();
   transforms(x,y,r,t);
   if(r<=1)
      positive++;
   if(r>1)
      negative++;
}
cout<<"Positive: "<<positive<<endl;
cout<<"Negative: "<<negative<<endl;
return 0;
}
```

Question 11: (5pts)

Now, let's compute π by using random numbers. First, we'll describe the math behind the technique. After that we'll give an expression for computing.

Suppose we draw a circle inside our square. Perhaps you recall that the radius of a circle is $A_c = \pi r^2$. If $r = 1$ then $A_c = \pi$ and the sides of the square that this circle is inside of would have length 2. Since the area of a square is $A_s = r^2$ where r is the length of a side of the square, the area of this square $A_s = 4$. We can compute the ratio of the area of the circle to the area of the square as $\frac{A_c}{A_s} = \frac{\pi}{4}$ and therefore compute $\pi = 4\frac{A_c}{A_s}$.

If our random points are uniformly distributed across the square then the ratio of the number of points inside the circle to the total number of points will be the same as the ratio of the areas (if we have enough random points).

Putting it all together, that means that we can estimate π by computing $4\frac{n_c}{n}$ where n is the number of points you generate and n_c is the number of those points that are inside the circle.

Write a program that uses the two functions you created to compute π by generating 100,000 points and counting how many are inside the circle. Be sure to seed the random number generator with the time.

Once you have it working paste your entire program below.

Answer:

```
// Joseph Mulray Lab 7
#include <iostream>
#include <string>
#include <cstdlib>
#include <cmath>
using namespace std;

void transforms (float x, float y, float &a, float &b)
{
  a=sqrt((x*x) + (y*y));
  b=atan(y/x);

}

float number()
{
   int temp;
   float a;
   temp = (rand() % 2000-1000);
   a = float(temp /1000.0);
```

```
    return a;
}

int main()
{
srand(time(NULL));
int positive=0;
int negative=0;
int total=100000;
float pi=0;
float r,t;

for(int c=1;c<=total;c++)
{
   float x=number();
   float y=number();
    transforms(x,y,r,t);
    if(r<1)
    {
        positive++;
    }
    if(r>1)
    {
        negative++;
    }
}
```

```
cout<<"Positive: "<<positive<<endl;
cout<<"Negative: "<<negative<<endl;

pi= (float) 4*positive /total;
cout<<"Pi: "<<(pi)<<endl;


return 0;

}
```