Computer Programming

Perl Programming Assignment #2, 2011

Brief:

You are to write a game program that is centered around the guessing of numbers.

Detail:

Your computer program should begin by making up a random number (more on this later). It should then ask the user to guess the number. Any guess from the user will have one of three states:

- Too low
- Correct
- Too high

Your program should then inform the user which state his/her guess falls in to. Use a loop to repeat this process until the user gets the right number. Since the number guessing range is from 1-1000 the user may get bored and wish to exit the game. If they 'guess' **-1** then the program should terminate, printing out how many guesses they made and what the number was.

If they guess correctly, the program should display a message to that effect, the number of guesses that were made and then exit.

Further below shows a sample of how the program might look. You are not required to produce identical output but attractiveness on screen gains more marks, as does the use of comments within the program to explain what individual pieces of code do.

Due Date:

20110331, 15:15

Submission Mechanism:

Paper & electronic program submission. On paper you should submit:

- Cover sheet
- Flow chart(s)
- Program listing
- Screen grabs showing program operation including test data

Send an email to **fachtna@fachtnaroe.net** with the subject line **PRG: #2**

- put your program in the body of the message
- attach one OpenOffice Writer file containing the program (in courier-style font), scans of your flow chart(s), sample screen grabs, gedit-printed code and at least one photograph of you either coding or running your program
- attach a desktop capture move (.ogv format) of your program being tested.

NOTE: Flow charts should be completed and marked 'FINAL' by Fachtna before coding your program.

Below is the output of a sample program in operation. This is an indicator of a minimum level of functionality required.

Sample Run:

```
I have chosen a number in the range 1 to 1000. See can you guess it.
Your guess?
254
Too Low. Try Again.
Your Guess?
567
Too High. Try Again.
Your guess?
512
Too Low. Try Again.
Your Guess?
550
Correct!!! In only 4 guesses.
Play again? Y
I have chosen a number in the range 1 to 1000. See can you guess it.
Your guess?
700
Too High. Try Again.
Your guess?
-1
You have given up after 1 guess. The number was 299. Goodbye.
```

Other features should be added to the program to improve its operation and presentation.

Random Numbers & Computers

This sample shows how poor computers naturally are at choosing random numbers. It seems only humans and the weather are truly random – and even we can be predicted. This program shows the same 'random' number being generated each time by the 'rand' function provided by the C programming language.

```
-bash-2.05b$ gcc a2.c
-bash-2.05b$ ./a.out
1804289383
-bash-2.05b$ ./a.out
1804289383
-bash-2.05b$ ./a.out
1804289383
```

The source code of this program is here:

```
#include <stdio.h>
main ()
{
 int x;
 x = rand (1000);
 printf ("%d\n", x);
}
```

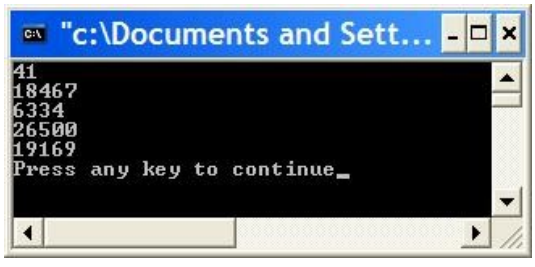
This program utilizes the **rand** function to generate it's 'random' function. As you can see above the number is not random – it's the same every time. That's because random number generators generally have to be 'seeded' before use.

This program tries to generate 5 'random' numbers:

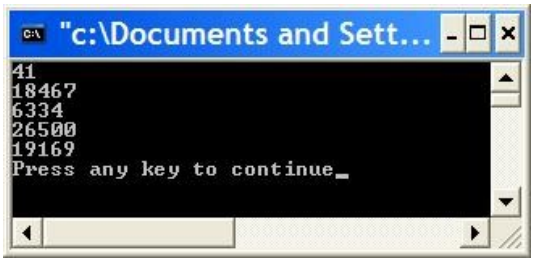
```
#include <stdio.h>
#include <stdlib.h>

int main()
{
 int i;
 for (i = 0; i < 5; i++)
 {
  printf("%d\n",rand());
 }
}
```

But each time produces:

Run 1



Run 2



'Seeding' The Random number generator:

If the program is modified to seed the random number generator with a value from the computer clock a near-random number can be generated. These numbers are known as pseudo-random. This line achieves this seeding process:

```
srand( (unsigned)time( NULL ) );
```

So that this program...

```
#include <stdio.h>
#include <time.h>
main ()
{
 int x;
 srand( (unsigned)time( NULL ) );
 x = rand ();
 printf ("%d\n", x);
}
```

yields this output:

```
-bash-2.05b$ gcc a2.c
-bash-2.05b$ ./a.out
404737816
-bash-2.05b$ ./a.out
1175092725
-bash-2.05b$ ./a.out
262886088
```

Perl random numbers

Random numbers in perl are a little easier as the seeding is automatic. This code generates a random decimal number between 0 and 1:

```
#!/usr/bin/perl

$a_number = rand ();
print $a_number . "\n";
```

To specify the range (other than 0 to 1) that you want the number to be in, put an upper value in the brackets, like this:

```
$a_number = rand (10);
```

This will generate a decimal value between 0 and 9. To push this value into the range of 1 to 10, add 1 like this:

```
$a_number = rand (10)+1;
```

Finally, to get an integer modify the program to use the int() function:

```
$a_number = int (rand (10))+1;
```