EECS 10: Assignment 4

Prof. Rainer Doemer

October 15, 2008

Due Monday 10/27/2006 12:00pm

1 Guess the Number [20 points]

Write a program that plays the game of ”guess the number”. In this game, the computer ”thinks” of a random number between 0 and a user-specified upper limit and the player has to guess this number. The computer will help the player by giving hints on whether the guessed number is less than or greater than the chosen number.

At the beginning, the computer asks the player for the upper bound for generating the random number. Your first line of output should display:

Enter the upper bound for the random number:

Once the user has entered the upper bound (say n=1000), your program chooses the number to be guessed by randomly selecting an integer in the range of 1 to n. The program then displays the following:

*“******Guessing Game******”*
I have selected a number in the range of 1 to 1000.
Can you guess the selected number?
Try No.1, please input the number:

The player then types a first guess. The program responds with one of the following according to the guess made:

- *Great!! You guessed it right! You have made X guesses.*
- *Your number was too low. Please try again!*
- *Your number was too high. Please try again!*

If the player does not succeed in guessing the number this round, then the program will prompt the player to guess it again as below(replace Y with the guess number):

Try No.Y, please input the number:

If the player’s guess is incorrect, your program should help the player to zero in on the correct answer by repeating the hints until the player finally gets the number right. At the end, display the number of guesses in total the player has made (in the text above, replace X with the proper number of guesses the player has made in total).

To show that your program works correctly, play it once with the upper bound 100 (i.e. range from 1 to 100) and submit the output as your script file (`guess.script`). Please compile your C code using **-ansi -Wall** options as below to specify ANSI code with all warnings:

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

You should submit your program code as file **guess.c**, a text file **guess.txt** briefly explaining how you designed your program, and a typescript **guess.script** which shows that you compile your program and run it. Try to guess a number between 1 to 200 (set the upper bound to 200).

HINT
To generate the initial 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 function 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 defined in 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 after the player inputs the upper bound *n*:

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

/* generate the random number in the range 0 to (n-1) */
int randomNumber = rand() % n;
```

Here, `n` specifies the upper bound of the range in which the random number will be generated, and `randomNumber` is the integer variable which is assigned the generated random number.

2 Time Value of Government Bailout Payment [20 points]

The government announced plans for a massive and unprecedented federal bailout of the US banking system recently. Given the value of the debt, annual percentage rate (APR) and yearly pay back, write a C program that computes and prints the interest paid and the remaining balance at the end of each yearly cycle. The program should continue printing the yearly values until the debt is paid off (i.e. the remaining balance becomes zero).

Your program should ask the user for the value of the debt, APR (in percent) and the yearly payment as input in the beginning.
For example, if the value of the debt = $60,000, APR = 4.5% (floating point value 4.5), and yearly payment = $5000, then your program should look as follows:

```
Enter the value of the debt   : 60000
Enter the APR                 : 4.5
Enter the yearly payment      : 5000
```

```
Year         Balance        Interest         Payment      NewBalance
2008        60000.00         2700.00         5000.00        57700.00
2009        57700.00         2596.50         5000.00        55296.50
2010        55296.50         2488.34         5000.00        52784.84
2011        52784.84         2375.32         5000.00        50160.16
2012        50160.16         2257.21         5000.00        47417.37
2013        47417.37         2133.78         5000.00        44551.15
2014        44551.15         2004.80         5000.00        41555.95
2015        41555.95         1870.02         5000.00        38425.97
2016        38425.97         1729.17         5000.00        35155.14
2017        35155.14         1581.98         5000.00        31737.12
2018        31737.12         1428.17         5000.00        28165.29
2019        28165.29         1267.44         5000.00        24432.73
2020        24432.73         1099.47         5000.00        20532.20
2021        20532.20          923.95         5000.00        16456.15
2022        16456.15          740.53         5000.00        12196.68
2023        12196.68          548.85         5000.00         7745.53
2024         7745.53          348.55         5000.00         3094.07
2025         3094.07          139.23         3233.31            0.00
It will take $        88233.31 till 2025 to pay off this debt.
```

NOTE: For all dollar amounts and the APR value, print out exactly 2 digits after the decimal point. Also ensure that all the numbers in the output table line up nicely so that the decimal points are all at the same column position. Also, for the last year, the payment need only be as much as the remaining debt. In the example, the last years payment is $3233.31 instead of $5000.00.

The first column Year keeps count of the number of years as the remaining debt diminishes each yearly cycle.

The second column is the “balance” at the beginning of every year. For the very first year, the value of the debt becomes the “balance”. For the subsequent year, previous year’s “total” will become the “balance”.

The third column “Interest” is the interest accrued on the Balance at the end of each yearly cycle. It is calculated using the formula:

```
Interest = balance*(APR/100).
```

The fourth column is the “Payment” at the end of each yearly cycle. In our program, this will be a constant value that the user inputs at the beginning except for the last year. In the above example, this is $5000.00.
The fifth column is the “New Balance” at the end of each yearly cycle. It is calculated using the following formula:

```
New Balance = Balance + Interest - Payment
```

You should submit your program code as file **debt.c**, a text file **debt.txt** briefly explaining how you designed your program, and a typescript **debt.script** which shows that you compile your program and run it.
Use the following assumptions to test your program: the value of the US Government debt is **700 billion** dollars, the APR is **4.5%**, and the yearly payment by taxpayers is **15 billion** dollars.

Hint: Since 700 billion is a great number, we need to pay attention to the range of defined variables. Here we may use `double` for this great value. Also, make sure that you have sufficient room in the displayed table for all the necessary digits.

3 Bonus [5 Points]

Let’s assume that the government debt will be paid off by the taxpayers (who are the real government). Extend Part 2 above to calculate the payment of each taxpayer every year. Suppose that the number of the taxpayers increases by

0.5% every year, and this year's number of the taxpayers is 95 million. Use the same file as in Part2 and the same test value. Add an additional column to show the payment of each taxpayer for each year.

Hint: You may use `double` variables for this bonus work.

To submit, use the same files as in Part 2, i.e. **debt.c**, **debt.txt**, and **debt.script**. Just add your code lines for this bonus part in these files.

4 Submission

Submission for these files will be similar to previous weeks' assignments. The only difference is that you need to create a directory called **`hw4/`**. Put all the files for assignment 4 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.