EECS 10: Assignment 4

October 13, 2006

Due Monday 10/23/2006 12:00pm

1 Time Value of Money [20 points]

Given initial principal amount, monthly payment/deposit amount (monthly installment), APR (Annual Percentage Rate), number of years, write a C program that computes and prints the interest and total value at the end of every month.

Your C program should ask the values of principle amount, APR, monthly payment/deposit, and number of years as inputs at beginning. For example, if principal value = $1200, APR = 1.5% (Floating point value 1.5), monthly deposit = $20, and the number of years = 2. Then at the beginning, your program should display as below:

Enter the principle amount (in $): 1200
Enter the APR (in %): 1.5
Enter monthly payment/deposit (in $): 20
Enter number of years interest is computed over: 2

Then your program should compute and print following data:

```
no.  year/mo      balance   payment/deposit    interest    total

  1     1/01      1200.00        20.00         1.50      1221.50

  2     1/02      1221.50        20.00         1.53      1243.03

  3     1/03      1243.03        20.00         1.55      1264.58
  .
  .
  .
 22     2/10      1657.19        20.00         2.07      1679.26

 23     2/11      1679.26        20.00         2.10      1701.36

 24     2/12      1701.36        20.00         2.13      1723.49

Summary:

Total Months  Initial Balance Total Payment/Deposit Total Interest Total.Balance

    24          1200.00              480.00           43.49          1723.49
```

NOTE: For all floating point values, please only print out 2 digits after decimal point.

The first column is the ”Number of Months”. For example, if the number of years = 2, then this column will run from 1 to 24. If number of years = 5, then it runs from 1 to 60.

The second column is the year and the month. It starts with 1/01, which means the first year and the first month.

The third column is the “balance” at the beginning of every month. For the very first month, Principal Value becomes the “balance”. For the subsequent months, previous month’s “total” will become the “balance”.

The fourth column is the monthly payment/deposit which will be constant in this case.

The fifth column is the interest earned by the “balance”. The values in this column are computed as:
$interest = balance * (apr/100)/12$

The sixth column is the total amount at the end of the month. It is computed as
$total = balance + payment/deposit + interest$

Run your program and get results for the following two cases:

- Savings account example: Principal Value=$1200, APR=1.5%, monthly deposit=$20, number of years=5
- Mortgage example: Borrowed Amount=$-290000, APR=6.5%, monthly pament=$1800, number of years=10

Please submit `interest.c`, `interest.script`, and `interest.txt` files. In `interest.txt` file, please use no more than 6 sentences to explain your program design.

2 Guess the Number [20 points + 5 points (extra credit)]

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 something like:

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 0 to (n-1). The program then displays the following:

*“******Guessing Game******”*
I have selected a number in the range of 0 to 999.
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 1000 (i.e. range from 0 to 999) 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
```

Please submit `guess.c`, `guess.script`, and `guess.txt` files. File `guess.txt` should briefly explain your program design within 10 sentences.

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 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 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.

Bonus Problems [5 points]:

How many steps at most does it take for a "Good Guesser" to guess any random number between 0 and 999? Answer the question and explain why in `guess.txt` concisely.

3 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 **`hw4/`**. Put all the files for assignment 4 in that directory and run the **`/ecelib/bin/turnin`** command to submit your homework.