EECS 10: Homework 4

Emad Arasteh, Prof. Rainer Dömer
October 16, 2019

Due Wednesday, October 30, 2019 at 12:00 noon

Part 1: Paying off Credit Card Debt (20 points)

Given a credit limit, current balance, annual percentage rate (APR) and monthly payment, write a C program that computes and prints the interest paid and the remaining balance at the end of each monthly cycle. The program should continue printing the monthly values until the debt is paid off (i.e. the remaining balance becomes zero).

Your program should ask the user for the credit limit, balance on the card, APR (in percent) and the monthly payment as inputs in the beginning.
For example, if credit limit = $5000, balance = $2000, APR = 14.99% (floating point value 14.99), and monthly payment = $200, then your program output should look as follows:

```
Enter the credit limit        : 5000.00
Enter the balance on the card : 2000.00
Enter the APR                 : 14.99
Enter the monthly payment     : 200.00

Month  Balance      Interest     Payment      New Balance
 1     2000.00      24.98        200.00       1824.98
 2     1824.98      22.80        200.00       1647.78
 3     1647.78      20.58        200.00       1468.36
 4     1468.36      18.34        200.00       1286.71
 5     1286.71      16.07        200.00       1102.78
 6     1102.78      13.78        200.00        916.55
 7      916.55      11.45        200.00        728.00
 8      728.00       9.09        200.00        537.10
 9      537.10       6.71        200.00        343.81
10      343.81       4.29        200.00        148.10
11      148.10       1.85        149.95          0.00

It will take $2149.95 over 11 month 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.

The first column ‘Month’ keeps count of the number of months as the remaining debt diminishes each monthly cycle.

The second column ‘Balance’ is the balance on the credit card at the beginning of each monthly cycle. For the first month, the balance is the value input by the user at the beginning of the program. For subsequent months, the balance is calculated using the following formula:

Balance = ‘New Balance’ from the previous month.

The third column ‘Interest’ is the interest accrued on the ‘Balance’ at the end of each monthly cycle. It is calculated using the formula: *Interest = balance*(APR/100)/12.*

The fourth column is the ‘Payment’ at the end of each monthly cycle. In our program, this will be a constant value that the user inputs at the beginning. In our example, this is $200.

The fifth column is the ‘New Balance’ at the end of each monthly cycle. It is calculated using the following formula: *New Balance = Balance + Interest - Payment*

You should submit your program code as file **`creditcard.c`**, a text file **`creditcard.txt`** briefly explaining how you designed your program, and a typescript **`creditcard.script`** which shows that you compile your program and run it. Use the following inputs to test your program:
Credit limit : $5000
Balance : $2000
APR : 14.99
Payment : $200

Part 2: Blackjack (20 points)

This is a Blackjack program. The idea is to simulate Blackjack in order to make programming more fun! (Yes! Write your own game!)

Here is the overview of the implementation:
To simulate the shuffled stack of cards, we use a pseudo random number generator that generates a random number in the range of 1 to 13. This represents the cards numbered 1 through 10, plus the jack, queen and king, respectively. If the card number is 1 to 10, it directly represents the value of the card. If the card number is 11 to 13, the card represents the jack, queen, and king, which all have the face value 10.

Player’s round: The dealer draws an initial card for the player and shows it. The player then can choose to draw additional cards as many times as he wants. If his cards have a combined value of more than 21, he loses immediately. If the player decides not to draw any more cards, it’s the dealer’s turn.

The interface should look like the following:

```
**********************************
** Welcome to EECS10 BlackJack! **
**********************************
Your first card is:7
Do you want another card?
Type 1 for Yes, 0 for No:
1
Your next card is:8
Your combined value is:15
Do you want another card?
Type 1 for Yes, 0 for No:
1
Your next card is:9
Your combined value is:24
Sorry. You lose!
```

Dealer's round: The dealer draws his own cards until he reaches one of the following conditions:
If his combined value reaches more than 21, the dealer loses.
If his combined value is the same as the player's value, the dealer wins.
If his combined value is higher than the player's value, the dealer wins.
An example code is shown below:

```
**********************************
** Welcome to EECS10 BlackJack! **
**********************************
Your first card is:7
Do you want another card?
Type 1 for Yes, 0 for No:
1
Your next card is:8
Your combined value is:15
Do you want another card?
Type 1 for Yes, 0 for No:
0
Dealer draws another card.
Dealer's card is:10
Dealer's value is 10, you have 15.
Dealer draws another card.
Dealer's card is:4
Dealer's value is 14, you have 15.
Dealer draws another card.
Dealer's card is:10
Dealer's value is 24, you have 15.
Dealer loses. You win!
```

You should submit your program code as file **`blackjack.c`**, a text file **`blackjack.txt`** briefly explaining how you designed your program, and a typescript **`blackjack.script`** which shows that you compile and run your program. Please run it twice so that the script shows that you and the dealer win one time each.

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 the 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 numbers every time you run it.

To set the seed value, you have to use the function **srand()**, which is also defined in the header file `stdlib.h`. For the current time of the day, you can use the function **time()**, which is defined in the 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(NULL));
```

3. To simulate drawing a card from the shuffled deck, use the following statement:

```
/* draw a random card */
 card = rand() % 13 + 1;
```

The integer variable 'card' then will have a random value in the range from 1 through 13.

3. Bonus Problem [5 Points]

Extend the blackjack program. To make the game more real, for each ace card (1), the player can choose the value to be either 1 or 11 for best interest. The decision can only be made once the card is issued and cannot be changed afterwards.

To submit, use the same files as in Part 2, i.e. **`blackjack.c`**, **`blackjack.txt`**, and **`blackjack.script`**.

4. Submission

Submission for the files will be similar to last week's assignment. Create a directory called hw4. Put all the files for assignment 4 in that directory and run the **`~eecs10/bin/turnin.sh`** command to submit your homework.