EECS 10: Assignment 4

Emad Arasteh, Prof. Rainer Doemer

October 16, 2018

Due Wednesday October 31, 2018 at 12:00pm

1 Rabbit Ecosystem and Demography [25 points]

Write a C program that analyzes rabbit and wolf population in an island and its influence on the grass area. Grass area in the island grows at a constant rate every year through rains, but is gradually depleted due to continuous consumption by rabbits. Rabbit population grows at a constant rate every year but is constrained by wolves attack. Wolf population grows at a constant rate every year but rapid decline occur in regular intervals through spread of epidemic diseases.

Your input section should look like the following:

```
Enter wolf population (initial): 10
Enter rabbit population (initial): 2300
Enter total grass area, initially fertile (in sq yards): 40000
Enter wolf annual growth rate (in percentage): 20
Enter rabbit annual growth rate (in percentage): 30
Enter grass area annual growth rate (in percentage): 5
```

These inputs along with below constraints will be used to calculate wolf population, rabbit population and available grass area for each year.

Wolf Population (2nd Column): Wolf population grows annually at the specific growth rate. Apart from this, every 5 years except year 1 wolf population decreases to half of previous year population due to widespread epidemic diseases. The following formula can be used to calculate wolf population where W_{i+1} and W_i are wolf populations for year $i+1$ and i respectively and g_w is growth rate of wolves:

$$W_{i+1} = \begin{cases} \frac{W_i}{2} & \text{if } i \neq 1 \text{ and } i\%5 = 1, \\ W_i * (1 + \frac{g_w}{100}) & \text{otherwise.} \end{cases}$$

Rabbit Population (3rd column): Rabbit population grows annually at the specific growth rate. Apart from this, each wolf kills 50 rabbits every year for their survival, apart from other prey. The following formula can be used to calculate rabbit population where R_{i+1} and R_i are rabbit populations for year $i+1$ and i respectively and g_r is growth rate of rabbits:

$$R_{i+1} = R_i * (1 + \frac{g_r}{100}) - W_i * 50$$

Available grass area (4th column): Grass area grows annually at the specific growth rate. Rabbits in this island consume grass continuously, and every year, each rabbit depletes grass in 1.2 sq yards of fertile grass area, and make it barren (due to rabbit consumption and insufficient rains for regrowth). The following formula can be used to calculate avaiable grass area where G_{i+1} and G_i are available grass area for year $i+1$ and i respectively and g_g is growth rate of grass area:

$$G_{i+1} = G_i * (1 + \frac{g_g}{100}) - R_i * 1.2$$

Your program should calculate and print the following data in a table format: Year, Wolf population, Rabbit population and Available grass area (in sq yards) for each year. Your final program output should look like this:

```
Enter wolf population (initial): 10
Enter rabbit population (initial): 2300
Enter total grass area, initially fertile (in sq yards): 40000
Enter wolf annual growth rate (in percentage): 20
Enter rabbit annual growth rate (in percentage): 30
Enter grass area annual growth rate (in percentage): 5
Year            Wolf population Rabbit population       Available Grass Area
0               10              2300                    40000.00
1               12              2490                    39240.00
2               14              2637                    38214.00
3               16              2728                    36960.30
4               19              2746                    35534.72
5               22              2619                    34016.25
6               11              2304                    32574.26
7               13              2445                    31438.18
8               15              2528                    30076.09
9               18              2536                    28546.29
10              21              2396                    26930.40
11              10              2064                    25401.72
12              12              2183                    24195.01
13              14              2237                    22785.16
14              16              2208                    21240.02
15              19              2070                    19652.42
16              9               1741                    18151.04
17              10              1813                    16969.39
18              12              1856                    15642.26
19              14              1812                    14197.18
20              16              1655                    12732.63
```

NOTES: Use a loop so that table is printed for 20 consecutive years as shown above. For year 0, only initial values are displayed. For available grass area, use double type variables, and 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. Use suitable number of tab characters in print statements to align the table output in same fashion as above. Wolf and rabbit population are integers, and fractional part can be ignored.

You should submit your program code as file **rabbit.c**, a text file **rabbit.txt** briefly explaining how you designed your program, and a typescript **rabbit.script** which shows that you compile your program and run it.

Your script file should contain output for following two scenarios.

```
A. Wolf population = 10, Rabbit population = 2300, Initial grass area = 40000
   Wolf growth rate = 20%, Rabbit growth rate = 30%, Grass growth rate = 5%
B. Wolf population = 15, Rabbit population = 6500, Initial grass area = 90000
   Wolf growth rate = 25%, Rabbit growth rate = 20%, Grass growth rate = 10%
```

2 Bonus: Minimum and maximum population [5 Points]

Calculate and print the minimum and maximum population of rabbits and wolves over these 20 years. Along with the minimum and maximum values, also print the corresponding years. Maintain variables for minimum and maximum wolf population and rabbit population seperately. For each case, maintain same number of variables for storing corresponding years. In your loop, update your maximum and minimum population by comparing with calculated population values. Also update corresponding year, when you update the population values.

Your program output should have an additional section (after the table in Problem 1), which should look like this:

```
Wolf population was minimum as 9 in year 16
Wolf population was maximum as 22 in year 5
Rabbit population was minimum as 1655 in year 20
Rabbit population was maximum as 2746 in year 4
```

To submit, use the same files as in Part 2, i.e. **rabbit.c**, **rabbit.txt**, and **rabbit.script**. Just add your code lines for this bonus part in these files. Your script file should have bonus section output for both scenarios as discussed in Part 1.

3 Guess the Age [15 points]

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

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

Enter the upper bound for the random age:

Once the user has entered the upper bound (say n=100), your program chooses the age 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 person with age in the range of 1 to 100.
Can you guess the selected age?
Try No.1, please input the age:

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.*
- *My selected person is older. Please try again!*
- *My selected person is younger. 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 age:

If the player's guess is incorrect, your program should help the player to zero in on the correct age by repeating the hints until the player finally gets the age 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 age 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 guess.c –ansi –Wall –o guess
```

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. Your script file should have output for guessing 2 ages with different upper age limits (100 and 120).

HINTS:
To generate the initial random age, 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 age in the range 0 to (n-1) */
int random_age = rand() % n;
```

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

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 **`~eecs10/bin/turnin.sh`** command to submit your homework.