

Practice 4 (2015/10/6)

Interactive Puzzle Game

1. Random Number Function

A. Random Number Generator

☞ Use *rand()* function to generate a random number.

B. Seed Number Initialization

☞ Random number generator is implemented by an algorithm that is initialized by a seed number. If you do not offer two different seed numbers to initialize random number generator for two runs, it will generate the same number sequence for two runs. To generate different number sequence in different runs, you have to initialize the generator with different seed numbers for different runs.

```
C:\WINDOWS\System32\WindowsPowerShell\v1.0\Powershell.exe
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void)
{
    time_t now;
    time(&now);
    printf("Now time is %s.\n",asctime(localtime(&now)));

    printf("random number 1 = %d\n",rand());
    printf("random number 2 = %d\n",rand());

    return 0;
}
```

```
C:\WINDOWS\System32\WindowsPowerShell\v1.0\Powershell.exe
C:\Users\boris\Documents\GitHub\CS1188\Practice_4> ./rand1
Now time is Sun Sep 25 15:44:02 2016
.
random number 1 = 1481765933
random number 2 = 1085377743
C:\Users\boris\Documents\GitHub\CS1188\Practice_4>
```

```
C:\WINDOWS\System32\WindowsPowerShell\v1.0\Powershell.exe
C:\Users\boris\Documents\GitHub\CS1188\Practice_4> ./rand1
Now time is Sun Sep 25 15:45:36 2016
.
random number 1 = 1481765933
random number 2 = 1085377743
C:\Users\boris\Documents\GitHub\CS1188\Practice_4>
```

☞ We run this program at different time but obtain the same random sequence.

☞ Use *srand()* and *time()* to initialize the generator with different seed numbers.

```

MINGW32/c/cygwin64/home/boris/CS1188/Practice_4
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void)
{
    time_t now;
    time(&now);
    printf("Now time is %s.\n", asctime(localtime(&now)));

    srand(time(NULL));

    printf("random number 1 = %d\n", rand());
    printf("random number 2 = %d\n", rand());

    return 0;
}

```

```

MINGW32/c/cygwin64/home/boris/CS1188/Practice_4
boris@boris-Novas MINGW32 /c/cygwin64/home/boris/CS1188/Practice_4
$ ./rand2
Now time is Sun Sep 25 16:14:30 2016
random number 1 = 575805069
random number 2 = 547188812

```

```

MINGW32/c/cygwin64/home/boris/CS1188/Practice_4
boris@boris-Novas MINGW32 /c/cygwin64/home/boris/CS1188/Practice_4
$ ./rand2
Now time is Sun Sep 25 16:15:30 2016
random number 1 = 1434931499
random number 2 = 386217541

```

☞ We use time as different seed numbers and then different random sequences are produced.

2. First Application of Array

A. Data and array index

☞ We can use data as the array index to represent some properties of data. For instance, 10 decimal digits (0 to 9) comprise all integers. If we want to know if any two digits of a number are the same. We can declare an array `used[10]` to imply the usage of each decimal digit.

used[10]	0	1	2	3	4	5	6	7	8	9	2	4	9	2	8	4
used[10]	0	0	1	0	0	0	0	0	0	0	2	4	9	2	8	4
used[10]	0	0	1	0	1	0	0	0	0	0	2	4	9	2	8	4
used[10]	0	0	1	0	1	0	0	0	0	1	2	4	9	2	8	4
used[10]	0	0	1	0	1	0	0	0	0	0	2	4	9	2	8	4
used[10]	0	0	1	0	1	0	0	0	0	0	2	4	9	2	8	4

```

MINGW32/c/cygwin64/home/boris/CS1188/Practice_4
#include <stdio.h>

int main(void)
{
    int num;

    printf("Please input a number to check the repeated occurrence of same digit number: ");
    fflush(stdout); // in mingw32, sometimes printf cannot print out message before scanf. Force to flush out
    scanf("%d",&num);

    int tmpVal = num, digitAry[10] = {0,0,0,0,0,0,0,0,0,0};

    for (; tmpVal > 0; ) {
        int digNum = tmpVal - (tmpVal/10) * 10;
        if (digitAry[digNum] == 1)
            printf("Digit number %d repeats at least twice in number %d\n",digNum, num);
        digitAry[digNum] = 1;
        tmpVal = tmpVal/10;
    }

    return 0;
}

```

3. Problem 1

A puzzle game for two players is described as follows. Two players design their own 4-digit number and two digits with the same number are not allowed. For instance, 1234 is a valid number while 1224 is invalid. The first digit can be 0. After setting their own number, each player starts to guess the designed number by the other player in turn. The guess number cannot contain the same digit number neither. Assume that numbers 3284 and 5901 are the numbers set by players 1 and 2. Initially player 1 guess a number 2490, then player 2 has to answer 2B to player 1. Then it is player 2's turn. If player 2 guess a number of 5289, player 1 has to answer 1A1B to player 2. Each player has to remember the numbers he has guessed and their related answers, analyze them and determine a new number for next guess. The one who first makes a right guess is the winner. Design a program to randomly generate a 4-digit integer and keep it in computer's mind. Then start a repeated procedure to let users input a number and report the outcome of current guess until the user gets a guess of 4A. The program has to report an invalid guess number, e.g., 4349 or 43a1. The program also has to report the error status for an invalid guess number.

This is the case that input number is not a 4-digit number.

```

MINGW32/c/cygwin64/home/boris/CS1188/Practice_4

boris@boris-Novas MINGW32 /c/cygwin64/home/boris/CS1188/Practice_4
$ ./practice4
Now time is Sun Sep 25 18:54:05 2016

Please input a 4-digit integer. Four digits must be four different numbers.
Your 4-digit number: 23df
You have to input a 4-digit number.
Press Enter to continue!

```

This is the case that two digits of the input number are the same.

```

MINGW32/c/cygwin64/home/boris/CS1188/Practice_4

Your 4-digit number: 2344
Your input number is 2344.          Two digits of the input number are the same. Retry again!
Press Enter to continue!

```

This is the case that the first character is a non-numerical character.

```

MINGW32:/c/cygwin64/home/boris/CS1188/Practice_4
Your 4-digit number: we232wta
Your input does not begin with a numerical character
Press Enter to continue!

```

This is the case that the number is not a 4-digit number.

```

MINGW32:/c/cygwin64/home/boris/CS1188/Practice_4
Your 4-digit number: 12345
You have to input a 4-digit number.
Press Enter to continue!

```

Repeat query operation and report the outcome of current guess until the player gets a right guess. After that ask the player if he wants to play again.

```

MINGW32:/c/cygwin64/home/boris/CS1188/Practice_4
Your 4-digit number: 1234
Your input number is 1234.      You got 0A3B.
Your 4-digit number: 9087
Your input number is 9087.      You got 0A0B.
Your 4-digit number: 6541
Your input number is 6541.      You got 1A1B.
Your 4-digit number: 2351
Your input number is 2351.      You got 1A3B.
Your 4-digit number: 5321
Your input number is 5321.      You got 0A4B.
Your 4-digit number: 2513
Your input number is 2513.
Congratulations! After 6 trials, you got the right number!
Do you want to play again? (y/n)

```

If the player enters 'y', the program generates another 4-digit number and starts the game again.

```

MINGW32:/c/cygwin64/home/boris/CS1188/Practice_4
Please input a 4-digit integer. Four digits must be four different numbers.
Your 4-digit number: |

```

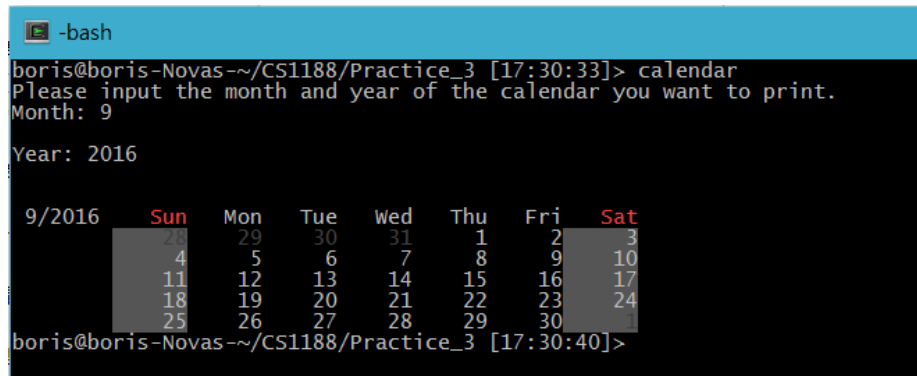
Main features: Since we have not introduced the function of C language, you can write all statements in the *main()* function. The entire program contains the following flow.

- (a) A 4-digit random number generator. You have to get current time as the seed number of *rand()* function to make sure the random numbers will look different in each run.
- (b) Read a number from the player and check if it is a 4-digit number. You can use the code in Practice 3.
- (c) Check if any two digits of the number are the same.
- (d) Compare the user-input number with the random number to determine how many *As* and *Bs* the use gets.
- (e) If the player gets a 4 *As*, congratulations to him; else return to (b) to start another guess.

(f) If the player gets a 4 As, ask the player whether he wants to play again. If yes, return (a) to produce a new 4-digit random number; else exit the game.

4. Problem 2: ANSI/VT100 Control Sequences in printf

Now you are able to get an input from users and print an output to a monitor. You can design a program to print monthly calendar with user's requested month and year. If the first day of the month is not on Sunday then you have to use the last several days of previous month to fill the empty columns of the first week. If the last day of the month is not on Saturday then you have to use the first several days of next month to fill the empty columns of the last week. Print the days of previous and next month in dark gray. In the first tab period of the first row, print the month and year given by the user. And then print the name from Sunday to Saturday. Print the names of Sunday and Saturday in red color. All days on Saturday and Sunday are printed in blinking. It looks like the following outcome.



```
-bash
boris@boris-Novas-~/CS1188/Practice_3 [17:30:33]> calendar
Please input the month and year of the calendar you want to print.
Month: 9
Year: 2016

9/2016   Sun   Mon   Tue   Wed   Thu   Fri   Sat
         28   29   30   31    1    2    3
         4    5    6    7    8    9   10
        11   12   13   14   15   16   17
        18   19   20   21   22   23   24
        25   26   27   28   29   30
boris@boris-Novas-~/CS1188/Practice_3 [17:30:40]>
```

5. Bonus

It's the computer's turn. Your computer does not want to simply answer how many As and Bs you get in a game. He or she wants to play a game with you. You design a 3-digit number and computer repeats to guess a 3-digit number. You have to answer how many As and Bs your computer gets. Then your computer will analyze all answers he or she has received from you and make a new guess until a 3A is gotten. A computer puzzle competition will be held three weeks later. The winner of the competition has to accept any new challenge before the end of this semester.

The program flow may look like this:

- Produce a 3-digit random number as my own secret number.
- Produce a 3-digit number for a guess and print my guessing number.
- Ask another player to input his guessing number.
- Print the outcome of the guessing number of another player.
- Receive the outcome of my current guess (input from another player).
- Analyze all previous outcomes and determine the number for next trial. Go

to (b).

6. **Further Reading**

Calendar time, CPU time, System time (terms in GNU), CPU time, System time (terms in computer organization)