Practice 4 (2015/10/6)
Interactive Puzzle Game

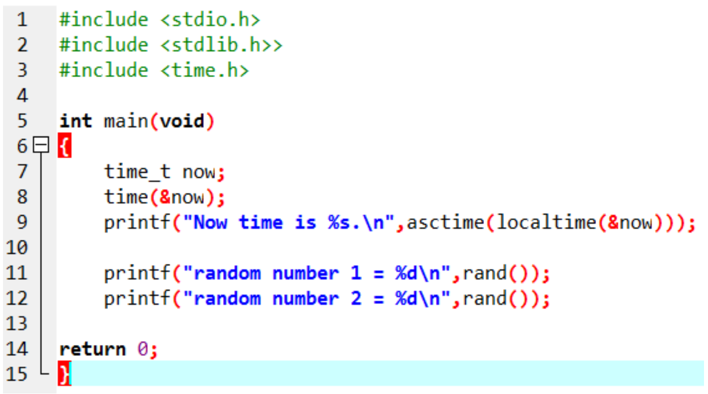
1. Random Number Function

A. Random Number Generator

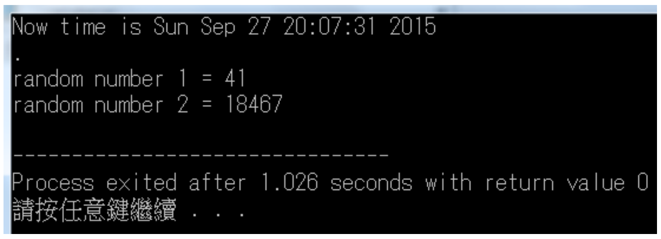
☞ Use *rand*() function to generate a random number.

B. Seed Number Initialization

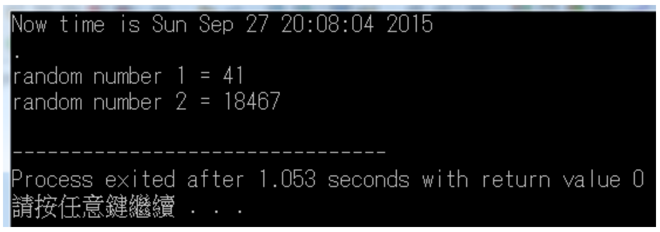
☞ Random number generator is implemented by an algorithm that is initialized by a seed number. If you do not offer two different seed numbers to initialize random number generator for two runs, it will generate the same number sequence for two runs. To generate different number sequence in different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



```c
#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;
}
```



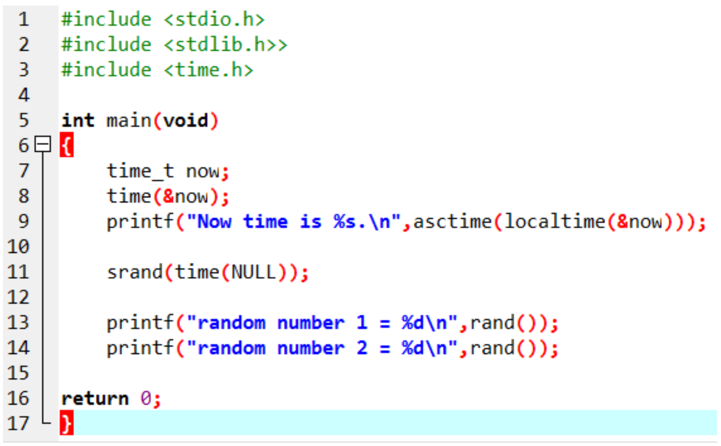
```
Now time is Sun Sep 27 20:07:31 2015
.
random number 1 = 41
random number 2 = 18467

--------------------------------
Process exited after 1.026 seconds with return value 0
請按任意鍵繼續 . . .
```



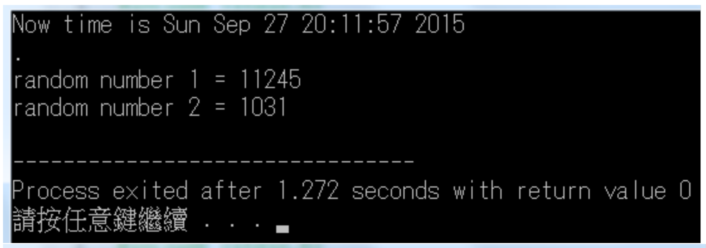
```
Now time is Sun Sep 27 20:08:04 2015
.
random number 1 = 41
random number 2 = 18467

--------------------------------
Process exited after 1.053 seconds with return value 0
請按任意鍵繼續 . . .
```

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

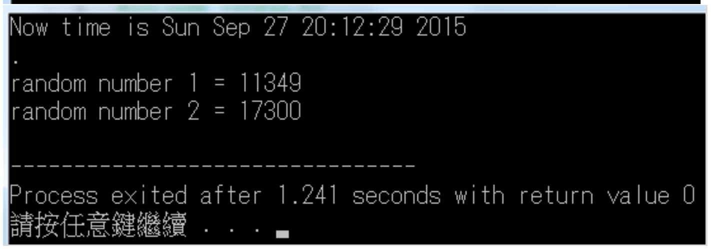
☞ Use *srand*() and *time*() to initialize the generator with different seed numbers.



```c
#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;
}
```



```
Now time is Sun Sep 27 20:11:57 2015
.
random number 1 = 11245
random number 2 = 1031

--------------------------------
Process exited after 1.272 seconds with return value 0
請按任意鍵繼續 . . .
```



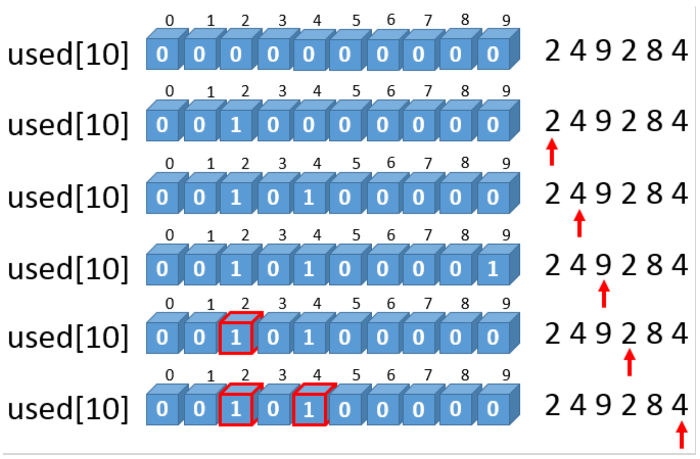
```
Now time is Sun Sep 27 20:12:29 2015
.
random number 1 = 11349
random number 2 = 17300

--------------------------------
Process exited after 1.241 seconds with return value 0
請按任意鍵繼續 . . .
```

☞ 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.

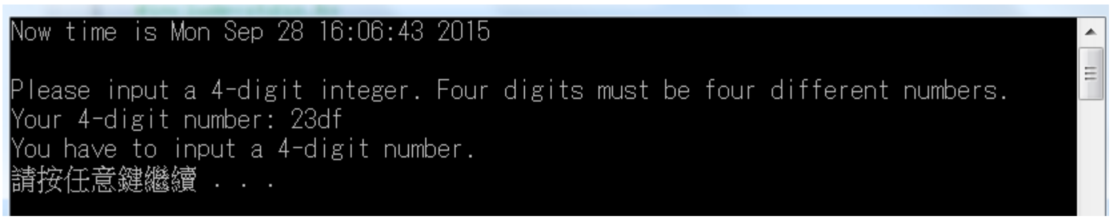


used[10] 0 0 0 0 0 0 0 0 0 0 2 4 9 2 8 4
used[10] 0 0 1 0 0 0 0 0 0 0 2 4 9 2 8 4
used[10] 0 0 1 0 1 0 0 0 0 0 2 4 9 2 8 4
used[10] 0 0 1 0 1 0 0 0 0 1 2 4 9 2 8 4
used[10] 0 0 1 0 1 0 0 0 0 0 2 4 9 2 8 4
used[10] 0 0 1 0 1 0 0 0 0 0 2 4 9 2 8 4
0 1 2 3 4 5 6 7 8 9

3. Problem

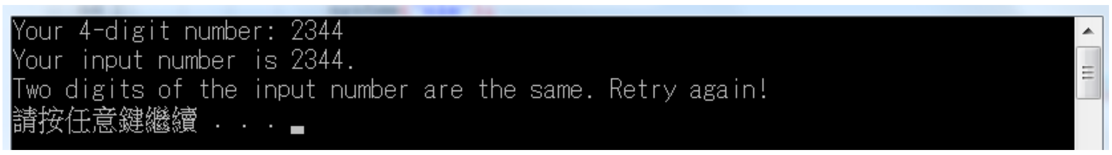
A puzzle game for two players is described as follows. Two players design their own 4-digit number and two digits with the same number are not allowed. For instance, 1234 is a valid number while 1224 is invalid. After setting their own number, each player starts to guess the designed number by the other 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.



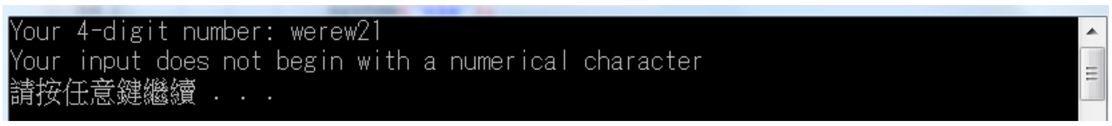
```
Now time is Mon Sep 28 16:06:43 2015

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.
請按任意鍵繼續 . . .
```

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



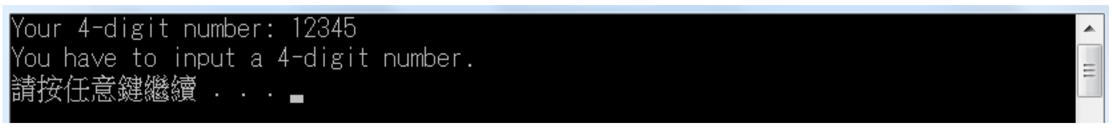
```
Your 4-digit number: 2344
Your input number is 2344.
Two digits of the input number are the same. Retry again!
請按任意鍵繼續 . . .
```

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



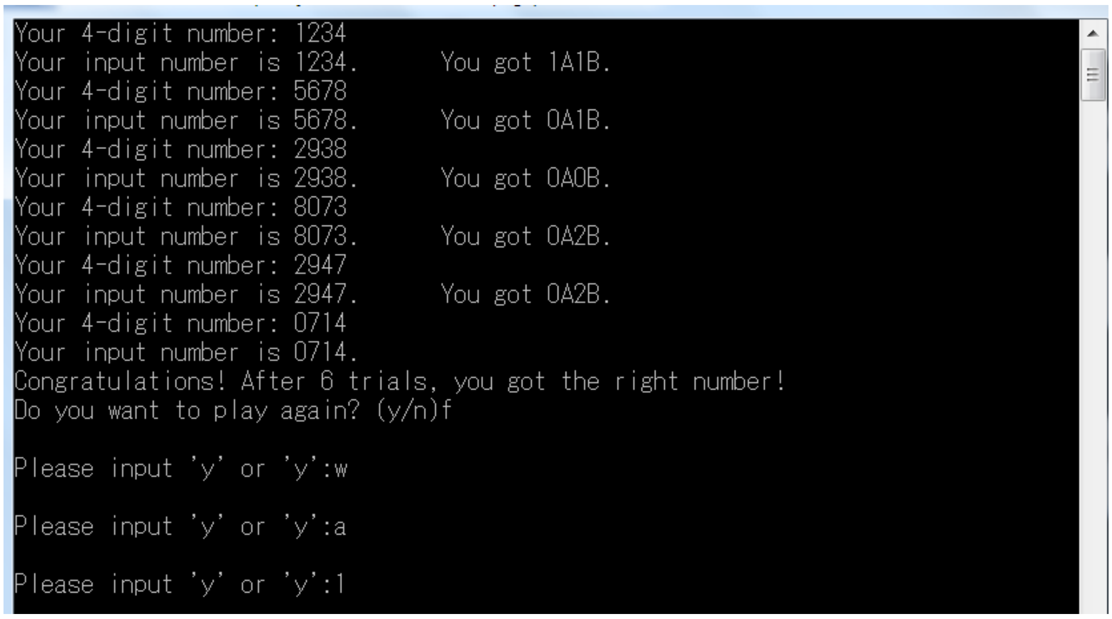
```
Your 4-digit number: werew21
Your input does not begin with a numerical character
請按任意鍵繼續 . . .
```

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



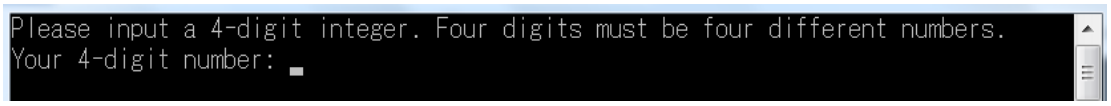
```
Your 4-digit number: 12345
You have to input a 4-digit number.
請按任意鍵繼續 . . .
```

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.



```
Your 4-digit number: 1234
Your input number is 1234.      You got 1A1B.
Your 4-digit number: 5678
Your input number is 5678.      You got 0A1B.
Your 4-digit number: 2938
Your input number is 2938.      You got 0A0B.
Your 4-digit number: 8073
Your input number is 8073.      You got 0A2B.
Your 4-digit number: 2947
Your input number is 2947.      You got 0A2B.
Your 4-digit number: 0714
Your input number is 0714.
Congratulations! After 6 trials, you got the right number!
Do you want to play again? (y/n)f

Please input 'y' or 'y':w

Please input 'y' or 'y':a

Please input 'y' or 'y':1
```

If the player enter 'y', the generate another 4-digit number and start the game again.



```
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 *A*s and *B*s the use gets.

(e) If the player gets a 4 *A*s, congratulations to him; else return to (b) to start another guess.

(f) If the player gets a 4 *A*s, 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. Bonus

It's the computer's turn. Your computer does not want to simply answer how many *A*s and *B*s 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 *A*s and *B*s your computer gets. Then your computer will analyze all answers he or she has received from you and make a new guess until a 3*A* 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:

(a) Produce a 3-digit random number as my own secret number.

(b) Produce a 3-digit number for a guess and print my guessing number.

(c) Ask another player to input his guessing number.

(d) Print the outcome of the guessing number of another player.

(e) Receive the outcome of my current guess (input from another player).

(f) Analyze all previous outcomes and determine the number for next trial. Go to (b).

5. Further Reading

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