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.

   ☞ 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.
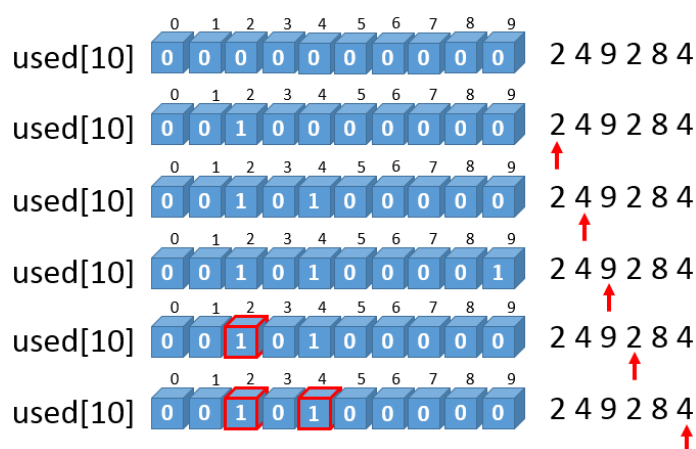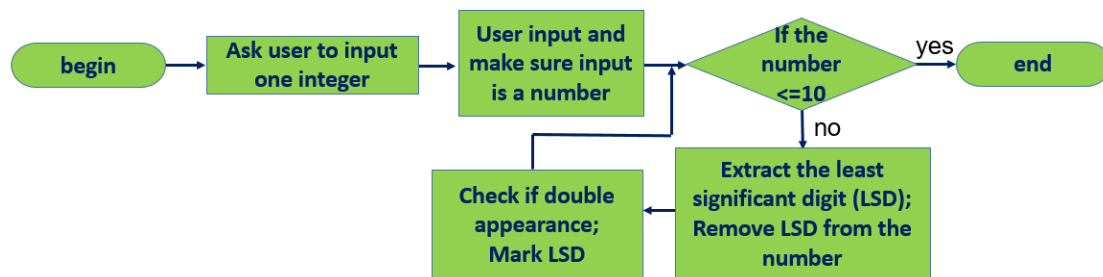
```
MINGW32:/c/cygwin64/home/boris/CS1188/Practice_4

#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.

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

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

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

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.

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

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

4

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.**