


Assignment 3 [9 marks]

Lucky Number for Date and Digital Number Segmentation

Due: 10th Dec. 2023, 23:59 [Week 15]

1. This assignment contains one question with **TWO** parts. You are required to submit a complete C++ program (.cpp only) for each part on **PASS** before the due date.
2. You can submit as many times as you want before the deadline, and we will grade your latest version.
3. Only a small set of test cases is visible for the testing, and a more comprehensive set will be used for grading. In other words, passing all the visible test cases does not mean you can get full marks for this assignment. Try your best to test your program thoroughly. Hidden test cases will not be released.
4. **The marking of each question is based on the percentage of the total test cases that your solution can pass.** If your submitted solution leads to a compilation error on PASS, zero marks will be given to that question, and no manual checking of your solution is provided in such a case.
5. **Late submissions will not be accepted.** ALL submissions should be on PASS.
6. **Plagiarism check** will be performed.



In this assignment, you will write a C++ program that can perform a simple UNO game (a kind of card game) under the following encoding scheme.

There are **two parts** in this encoding scheme. The first part aims to initialize a sequence of cards which will be used for playing the simple UNO game. The second part is playing the simple UNO game.

Part A. [4 mark] Initializing a Sequence of Cards

Your first task is to write a program that can initialize a sequence of cards, which is used for playing the simple UNO game. The program takes one integer from the user and it is the seed for the card sequence generation. As shown in the following Table1, the value of seed is from 1 to 20, and each seed corresponds to one specific sequence. Then it prints the entire sequence of cards. We provide a skeleton of code in the file [asg3_skeleton_partA.cpp](#), and you need to complete the program.

Now to begin, a card has three attributes: color, value and number. There are 4 colors with their corresponding values: Red with value 1, Yellow with value 2, Blue with value 3 and Green with value 4. There are 9 numbers: the integers from 1 to 9. A card is combined with one color and one number randomly, such as Red3, Yellow3, Blue7 and Green 1.

A class called **Card** is provided in the skeleton code, with the above attributes defined as private members. **Color** is a pointer to char, i.e., **char***, or equivalently cstrings. Some access function prototypes are also defined. There is also a prototype of the default constructor. You need to implement these access functions and the default constructor in order to complete the class definition of **Card**.

To initialize a sequence of cards, you need to implement one function. A sequence is simply an array of **Card** objects as you can see in **main()**. The **initSequence(Card* cardSeq, char colorName[][10], int* j, int* num)** is the function to initialize the sequence of cards. For one sequence, there are 6 cards. When you create each card, you should first set its value and its color according to the value as shown in Table1. Then, combining the color with the number, you can get a card. Note that the sequence is passed to the function as a pointer (**Card***), and you should not change this.

We provide a help function called **printSequence** which simply takes the sequence (again as a pointer) and prints each Card object sequentially. This is called in **main()** to check output of your code.

Notes:

1. Based on skeleton code, complete the class definition of Card.
2. You are not allowed to delete any part of the skeleton code.
3. Do not add / move attributes or member functions.
4. Finish all the default constructor or functions.
5. Table1 is shown below.

Seed No.	Card Sequence
1	Yellow9 Blue5 Yellow2 Blue1 Blue3 Yellow3
2	Yellow3 Blue3 Red4 Blue8 Yellow2 Yellow2
3	Red6 Blue2 Green7 Blue5 Yellow1 Yellow1
4	Green9 Green9 Yellow1 Green3 Red9 Yellow9
5	Blue2 Green6 Red3 Green1 Red7 Blue8
6	Blue5 Green5 Blue7 Red8 Red6 Blue7
7	Yellow8 Green3 Yellow1 Red5 Green5 Blue6
8	Red1 Green9 Green4 Yellow3 Green3 Blue5
9	Red5 Green7 Blue6 Yellow1 Blue2 Blue4
10	Green7 Red6 Yellow9 Blue7 Blue1 Blue3
11	Blue1 Red4 Green3 Blue5 Blue8 Blue2
12	Yellow4 Red1 Blue6 Green2 Yellow7 Blue1
13	Yellow6 Red9 Red8 Green9 Yellow6 Blue9
14	Red9 Red7 Green3 Green6 Red5 Blue8
15	Green3 Red5 Yellow6 Red4 Red4 Green7
16	Blue6 Yellow3 Red9 Red2 Green2 Green6
17	Blue8 Yellow1 Green2 Yellow9 Green1 Green5
18	Yellow2 Yellow8 Yellow5 Yellow6 Green9 Green4
19	Red5 Yellow7 Red8 Blue4 Blue8 Green3
20	Green7 Yellow4 Blue1 Blue2 Blue7 Green2

Sample Input and Output

Example 1

Microsoft Visual Studio Debug Console

```
Enter the seed for random number generation: 1
Yellow9 Blue5 Yellow2 Blue1 Blue3 Yellow3
```

Example 2

Microsoft Visual Studio Debug Console

```
Enter the seed for random number generation: 10
Green7 Red6 Yellow9 Blue7 Blue1 Blue3
```

Example 3

Microsoft Visual Studio Debug Console

```
Enter the seed for random number generation: 6
Blue5 Green5 Blue7 Red8 Red6 Blue7
```

Part B. [5 marks] Playing the Simple UNO Game

Extend your program in Part A, so it can play the simple UNO game with you as the only player. We provide a skeleton of code in the file [asg3_skeleton_partB.cpp](#), and you need to complete the program.

The game is follows. At first, the program asks the user to input the seed for the card sequence generation. It initializes a sequence of cards which are your cards, just like the basic level. Then the game starts.

In the first round, the default function in skeleton code will call the function you designed in part A and initialize following specific sequence of cards for computer (**[Red1 Green9 Green4 Yellow3 Green3 Blue5]**). Then the computer will show you the first card in its sequence.

In the second round, the program will ask you to enter two integers to represent the card you play: one is the value from 1 to 4 and the other is the number from 1 to 9. But there are three rules for your input. First, the card you play must within the card sequence you generated before the game starts. Second, this card will be removed from the sequence after you play it. Third, the card you play must be either the same color or number as the computer played in the last round. After you play the card, the program will print your remaining card sequence.

In the third round, the computer will play a card which also follows the three rules mentioned above. First, the card that computer plays must within the card sequence it generated at the first round. Second, this card will be removed from the computer's card sequence after the computer plays it. Third, the card that the computer plays must be either the same color or number as you played in the last round. You should design the logic code for computer to help it achieve this step: the computer should choose the **first** suitable card in its card sequence as its strategy.

In the fourth round, the process will be the same as the second round.

In the fifth round, the process will be the same as the third round.

.....

The game will end if one of these conditions are met: (1) There is no suitable card for you to play and you lose. (2) There is no suitable card for the computer to play and you win. (3) All the cards (yours and the computer's) are suitably played and get a draw.

Some sample runs of the game are shown below. Note your program output should be exactly the same as the sample output below, using the same seed.

Notes:

1. Refer to the Notes of Part A.
2. You may add new member functions to Card and new non-member functions to the program.
3. If there is no suitable card for you to play, please input 0 0 and then the game will end.
4. If there are some suitable cards for you to play, but you do not play any one of them, you will lose.

Sample Input and Output

Example 1

```
Microsoft Visual Studio Debug Console
Enter the seed for random number generation: 1
Your card sequence: Yellow9 Blue5 Yellow2 Blue1 Blue3 Yellow3
Computer's card sequence: Red1 Green9 Green4 Yellow3 Green3 Blue5
The computer plays Red1, please input two integers to represent the card you plan to play: 3 1
Game continues. Your remaining card(s): Yellow9 Blue5 Yellow2 Blue3 Yellow3
The computer plays Blue5, please input two integers to represent the card you plan to play: 2 9
Game ends. you lose
```

Example 2

```
Microsoft Visual Studio Debug Console
Enter the seed for random number generation: 19
Your card sequence: Red5 Yellow7 Red8 Blue4 Blue8 Green3
Computer's card sequence: Red1 Green9 Green4 Yellow3 Green3 Blue5
The computer plays Red1, please input two integers to represent the card you plan to play: 1 5
Game continues. Your remaining card(s): Yellow7 Red8 Blue4 Blue8 Green3
The computer plays Blue5, please input two integers to represent the card you plan to play: 3 8
Game ends. You win
```

Example 3

```
Microsoft Visual Studio Debug Console
Enter the seed for random number generation: 3
Your card sequence: Red6 Blue2 Green7 Blue5 Yellow1 Yellow1
Computer's card sequence: Red1 Green9 Green4 Yellow3 Green3 Blue5
The computer plays Red1, please input two integers to represent the card you plan to play: 1 6
Game ends. You win
```

Example 4

```
Microsoft Visual Studio Debug Console
Enter the seed for random number generation: 13
Your card sequence: Yellow6 Red9 Red8 Green9 Yellow6 Blue9
Computer's card sequence: Red1 Green9 Green4 Yellow3 Green3 Blue5
The computer plays Red1, please input two integers to represent the card you plan to play: 1 9
Game continues. Your remaining card(s): Yellow6 Red8 Green9 Yellow6 Blue9
The computer plays Green9, please input two integers to represent the card you plan to play: 4 9
Game continues. Your remaining card(s): Yellow6 Red8 Yellow6 Blue9
The computer plays Green4, please input two integers to represent the card you plan to play: 0 0
Game ends. you lose
```