Mini Project #1 **Due date: 17 December, 2017**
Fundamentals of Programming **Total points 20**

Simplified Solitaire Encryption Algorithm

In Neal Stephenson's novel **Cryptonomicon**, two of the main characters are able to covertly communicate with one another with a deck of playing cards (including the jokers) and knowledge of the Solitaire encryption algorithm, which was created (in real life) by Bruce Schneier.

For this project, we'll simplify the algorithm in several ways. For example, we'll be assuming that we have just two suits (say, hearts and spades) from a deck of cards, plus the two jokers, just to keep things simple. Further, let's assume that the values of the 26 suit cards are 1 to 26 (Ace to King of hearts, followed by Ace to King of spades), that the "A" joker is 27, and that the "B" joker is 28. Thus, 15 represents the 2 of spades.

Now that you've got the idea, note that because we are doing this in a computer, we can just use the numbers 1–28 and forget about the suits and ranks.

Task #1. Generate the Keystream Value

The hard part of Solitaire is the generation of the keystream values; they will be used to encrypt or decrypt messages. Here are the steps used in our variant of the algorithm, assuming that we start with a list of the values from 1–28 as described above:

- **Step 1.** Find the A joker (27). Exchange it with the card beneath (after) it in the deck, to move the card down the deck by one position. (What if the joker is the last card in the deck? Imagine that the deck of cards is continuous; the card following the bottom card is the top card of the deck, and you'd just exchange them.)
- **Step 2.** Find the B joker (28). Move it two cards down by performing two exchanges.
- **Step 3.** Swap the cards above the first joker (the one closest to the top of the deck) with the cards below the second joker. This is called a triple cut.
- **Step 4.** Take the bottom card from the deck. Count down from the top card by a quantity of cards equal to the value of that bottom card. (If the bottom card is a joker, let its value be 27, regardless of which joker it is.) Take that group of cards and move them to the bottom of the deck. Return the bottom card to the bottom of the deck.
- **Step 5.** Look at the top card's value (which is again 1-27, as it was in the previous step). Put the card back on top of the deck. Count down the deck by that many cards. Record the value of the NEXT card in the deck, but don't remove it from the deck. If that next card happens to be a joker, don't record anything. Leave the deck the way it is, and start again from the first step, repeating until that next card is not a joker.

The value that you recorded in the last step is one value of the keystream, and will be in the range 1–26, inclusive (to match with the number of letters in the alphabet). To generate another value, we take the deck as it is after the last step and repeat the algorithm. We need to generate as many keystream values as there are letters in the message being encrypted or decrypted.

Created by. Lester I. McCann (The University of Arizona).

Example

To make sense of the algorithm. Let's say that this is the original ordering of our half–deck of cards:

1 4 7 10 13 16 19 22 25 28 3 6 9 12 15 18 21 24 27 2 5 8 11 14 17 20 23 26

- **Step 1.** Swap 27 with the value following it. So, we swap 27 and 2:

  1 4 7 10 13 16 19 22 25 28 3 6 9 12 15 18 21 24 2 27 5 8 11 14 17 20 23 26

- **Step 2.** Move 28 two places down the list. It ends up between 6 and 9:

  1 4 7 10 13 16 19 22 25 3 6 28 9 12 15 18 21 24 2 27 5 8 11 14 17 20 23 26

- **Step 3.** Do the triple cut. Everything above the first joker (28 in this case) goes to the bottom of the deck, and everything below the second (27) goes to the top:

  5 8 11 14 17 20 23 26 28 9 12 15 18 21 24 2 27 1 4 7 10 13 16 19 22 25 3 6

- **Step 4.** The bottom card is 6. The first 6 cards of the deck are 5, 8, 11, 14, 17, and 20. They go just ahead of 6 at the bottom end of the deck:

  23 26 28 9 12 15 18 21 24 2 27 1 4 7 10 13 16 19 22 25 3 5 8 11 14 17 20 6

- **Step 5.** The top card is 23. Thus, our generated keystream value is the 24^{th} card, which is 11.

Self Test

Once done implementing the Solitaire algorithm, confirm the key stream value generated is 11 for the example deck given in the example. Use the deck at the end of the last step to generate the next keystream value. The answer is provided at the end of this document.

Task #2. Encoding using Solitaire

Now what do you do with all of those keystream values? The answer depends on whether you are encoding a message or decoding one.

To encode a message with Solitaire, remove all non-letters and convert any lower-case letters to upper-case (refer to ctype.h for utility functions.) Convert the letters to numbers (A=1, B=2, etc.). Use Solitaire to generate the same number of values as are in the message. Add the corresponding pairs of numbers, modulo 26. Convert the numbers back to letters, and you're done.

Example

Consider the message to be sent is this:

```
It's fun to program.
```

- Removing the non-letters and capitalizing gives us:

```
ITSFUNTOPROGRAM
```

- Next, convert the letters to numbers:

```
 I  T  S  F  U  N  T  O  P  R  O  G  R  A  M
 9 20 19  6 21 14 20 15 16 18 15  7 18  1 13
```

- Generate a sequence of 15 keystream values.

```
11  9 23  7 10 25 11 11  1 20 26 14  8  6  2
```

- Just add the two groups together pairwise. To get the modulo 26: If the sum of a pair is greater than 26, just subtract 26 from it. For example, 14 + 12 = 26, but 14 + 23 = 37 - 26 = 11.

```
   9 20 19  6 21 14 20 15 16 18 15  7 18  1 13
+ 11  9 23  7 10 25 11 11  1 20 26 14  8  6  2
-----------------------------------------------
  20  3 16 13  5 13  5 26 17 12 15 21 26  7 15
```

- And convert back to letters:

```
   T  C  P  M  E  M  E  Z  Q  L  O  U  Z  G  O
```

Task #3. Decoding the Message

Decryption is just the reverse of encryption. Start by converting the message to be decoded to numbers. Using the same card ordering as was used to encrypt the message originally, generate enough keystream values. Because the same starting deck of cards was used, the same keystream will be generated. Subtract the keystream values from the message numbers, again modulo 26. Finally, convert the numbers to letters and read the message.

Here's how the recipient would decrypt this message.

- Convert the encrypted message's letters to numbers.
- Generate the same keystream (by starting with the same deck ordering as was used for the encryption).
- Subtract the keystream values from the message numbers. To deal with the modulo 26 this time, just add 26 to the top number if it is equal to or smaller than the bottom number.

```
  20  3 16 13  5 13  5 26 17 12 15 21 26  7 15
- 11  9 23  7 10 25 11 11  1 20 26 14  8  6  2
-----------------------------------------------
   9 20 19  6 21 14 20 15 16 18 15  7 18  1 13
```

- Finally, convert the numbers to letters, and you're done.

```
   9 20 19  6 21 14 20 15 16 18 15  7 18  1 13
   I  T  S  F  U  N  T  O  P  R  O  G  R  A  M
```

Program Output

Sample run of your program should give similar output.

```
+---------------------+
| Solitaire Algorithm |
+---------------------+

Input Message. It's fun to program.

ENCRYPTION PASS ...

Cleaned Message.        I  T  S  F  U  N  T  O  P  R  O  G  R  A  M
Letters to Numbers.     9 20 19  6 21 14 20 15 16 18 15  7 18  1 13
Generated Keystreams. 11  9 23  7 10 25 11 11  1 20 26 14  8  6  2
Encoded Numbers.      20  3 16 13  5 13  5 26 17 12 15 21 26  7 15
Encoded Message.        T  C  P  M  E  M  E  Z  Q  L  O  U  Z  G  O

DECRYPTION PASS ...

Encoded Message.        T  C  P  M  E  M  E  Z  Q  L  O  U  Z  G  O
Letters to Numbers.    20  3 16 13  5 13  5 26 17 12 15 21 26  7 15
Generated Keystreams. 11  9 23  7 10 25 11 11  1 20 26 14  8  6  2
Decoded Numbers.        9 20 19  6 21 14 20 15 16 18 15  7 18  1 13
Decoded Message.        I  T  S  F  U  N  T  O  P  R  O  G  R  A  M
```

Hand-in

This is a TEAM project. Form one with atmost 2 (two) people (team with single member is also allowed). Designate one member as the "team leader", who will be responsible for the submission of the project.

Hand-in ONLY the files listed below at the appropriate location on the blackboard system at LMS. You should hand in a single compressed/archived file named Project_1_<team leader reg. No. XXX without angle brackets>.zip that contains the following files.

1. COMMENTED source files with authors information at the beginning (only .c and .h files).
2. CLEAR snapshot of your console (SNAPSHOT.png) to show that your program is working.
3. PLAIN text file (OUTPUT.txt) that includes a) authors information at the beginning, b) a brief explanation of the project, c) difficulties faced, and d) any comments, or suggestions.

Programs will be graded on the following criteria:

1. **Program Specifications / Correctness.** does it compile? Are there obvious errors? Are there subtle errors? (25%)
2. **Documentation.** Is your program consistently indented in a manner that reflects the structure of your code? Is it easy to read? Are there blank lines which break up the major sections of your code? (15%)

3. **Efficieny.** Is your program efficiently organized, or is there a lot of duplicated code? Does it look well-written, or barely finished? (5%)
4. **Project Specifications.** Does your program fulfill the basic requirements? Is it done? And what else does it do? (15%)
5. **In time.** (10%)
6. **Plagiarism.** (30%)

Honor code

The student should agree to the terms below; any infringements will result in minimum marks.

- will not cheat on project
- will not share solutions to the project; and
- will notify the instructor immediately if he or she becomes aware of any other group cheating

Some Tips

- Start early! There are lots of little things that need to be done to write this program.
- Make sure that you understand the modified Solitaire algorithm before you start writing the program.; you can't write a program to solve a problem you don't understand.
- Don't try to write the whole program at once; start small. For example, you're going to find the jokers. Write that method and test it. Then move on.
- You can check your program's work by hand. Take the given initial deck of cards, manually generate the first few keystream letters, and check that your program generated the same ones.
- Write an separate encoding and decoding methods for your program, so that your program can encode encrypted messages, as well as, decode decrypted messages.
- Exchange encrypted messages with your classmates for them to decrypt. This is to test your own encryption and decryption routines for any logical errors with the algorithm implementation is likely to appear in both routines. Without independent verification, you may think that your logically–flawed code is correct.

References

Solitaire Algorithm

Answer to the Self Test

- **Step 1.**

  23 26 28 9 12 15 18 21 24 2 1 27 4 7 10 13 16 19 22 25 3 5 8 11 14 17 20 6

- **Step 2.**

  23 26 9 12 28 15 18 21 24 2 1 27 4 7 10 13 16 19 22 25 3 5 8 11 14 17 20 6

- **Step 3.**

  4 7 10 13 16 19 22 25 3 5 8 11 14 17 20 6 28 15 18 21 24 2 1 27 23 26 9 12

- **Step 4.**

  14 17 20 6 28 15 18 21 24 2 1 27 23 26 9 4 7 10 13 16 19 22 25 3 5 8 11 12

- **Step 5.**

  The deck is the same as it was after Step 4. The 15^{th} card, the next keystream value, is 9.