Stacks

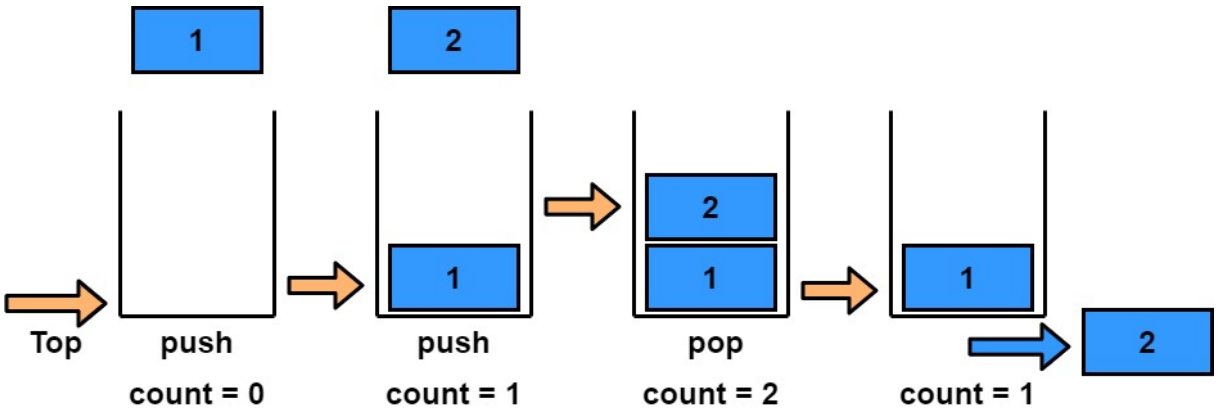
Purpose: The aim of this lab is to get familiar with stack data structures and basic operation on them.

Introduction

Stacks are the data structures that consist of a stack pointer and a list of items. Stack pointer indicates the top item of the stack and does not need to be a real pointer. In fact, a counter that counts the number of items in the stack can be used as stack pointer. Insertion, removal and element accessing operations can be applied only to the top of the stack.

Problem Statement

In this lab, you need to generate a stack structure to represent a deck of cards numbered between $(1, MAXSTACKSIZE)$. Then, you will shuffle the cards stored in the stack in two specific ways.



Lab Procedure

We highly recommend you to follow lab procedure without skipping any step and read each step thoroughly before you start.

1- Define maximum stack size and stack structure. (5 pts)

```
#define MAXSTACKSIZE 10
struct Deck{
   int val[MAXSTACKSIZE];
   int count;};
```

2- Implement "pop" and "push" operations for "Deck". (10 pts total - 5 pts each)

```
void push(struct Deck *top, int newVal)
void pop(struct Deck *top)
```

3- Implement "fillDeck" function that uses "push" function to fill a empty stack. "size" denotes the number of cards that will stored in the "Deck". The values (val) of cards stored in the stack should be in descending (from bottom to top, cf. Figure 1) order from $MAXSTACKSIZE$ to 1. (10 pts)

```
void fillDeck(struct Deck *top, int size)
```

4- Implement "riffle" function that rearranges the order of entries in the stack. The riffle procedure depicted in Figure 1 can be done by calling the "push" and "pop" functions $MAXSTACKSIZE * 2$ times (each). It is possible to reduce number of function calls and if you achieve to implement the "riffle" function using less than $MAXSTACKSIZE * 2$ function calls, you get bonus 10 pts from this section. (30 pts, +10 bonus pts) **Note: Your total score can not pass 100.**

```
void riffle(struct Deck *top)
```

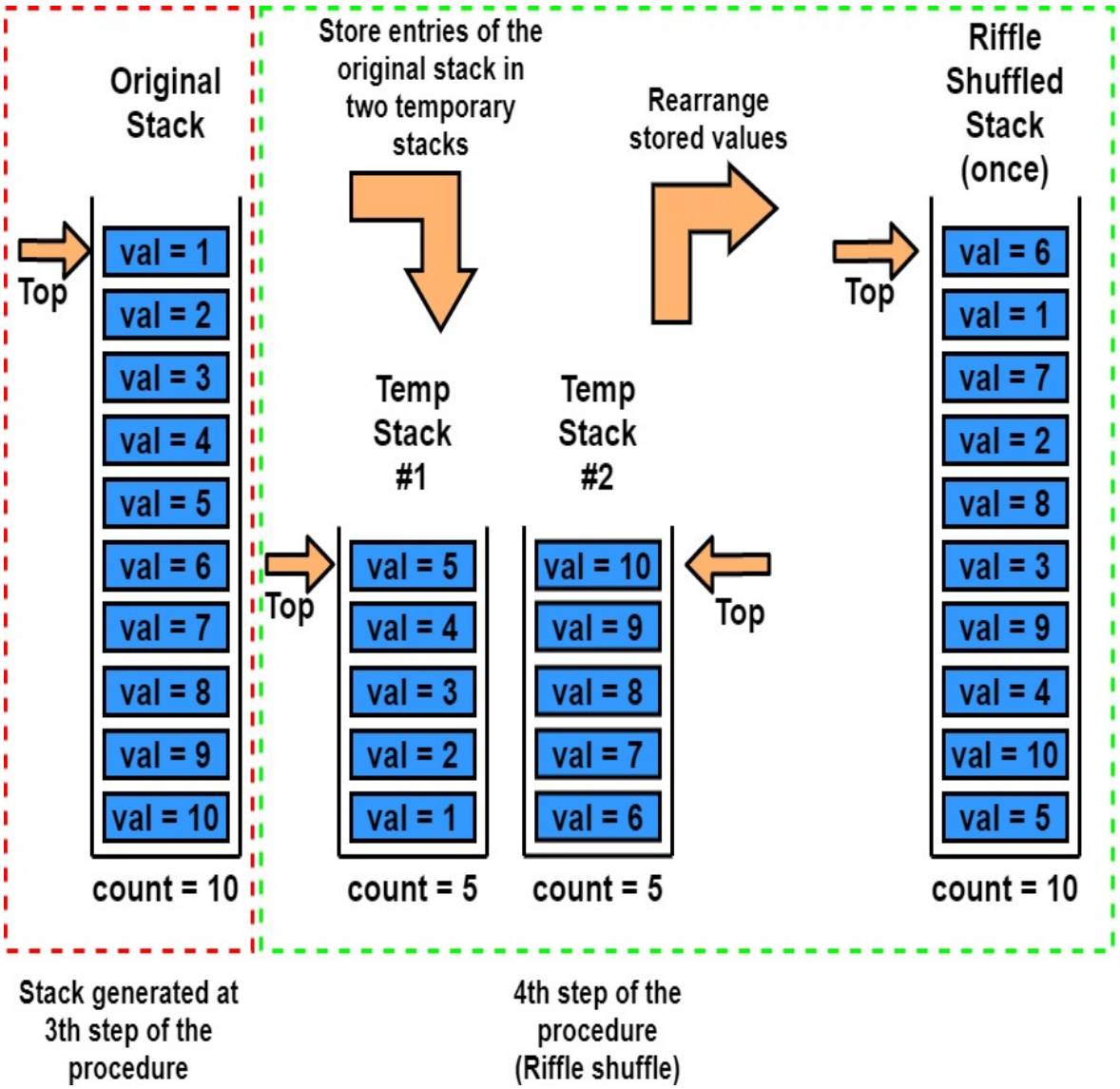


Figure 1: Riffle Shuffle Procedure

5- Implement Fisher-Yates shuffle algorithm. The procedure is depicted in Figures 2-5. (30 pts)

```
void fisheryates(struct Deck *top)
```

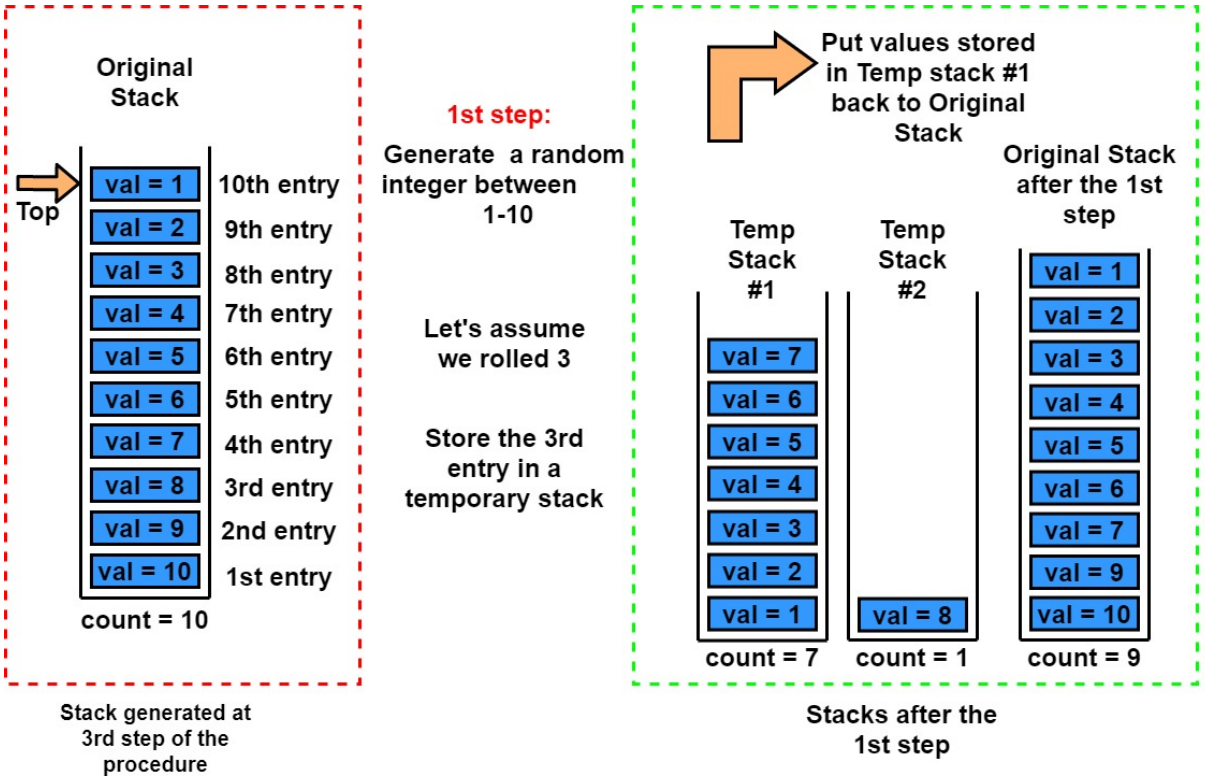


Figure 2: Fisher-Yates shuffle the 1st step

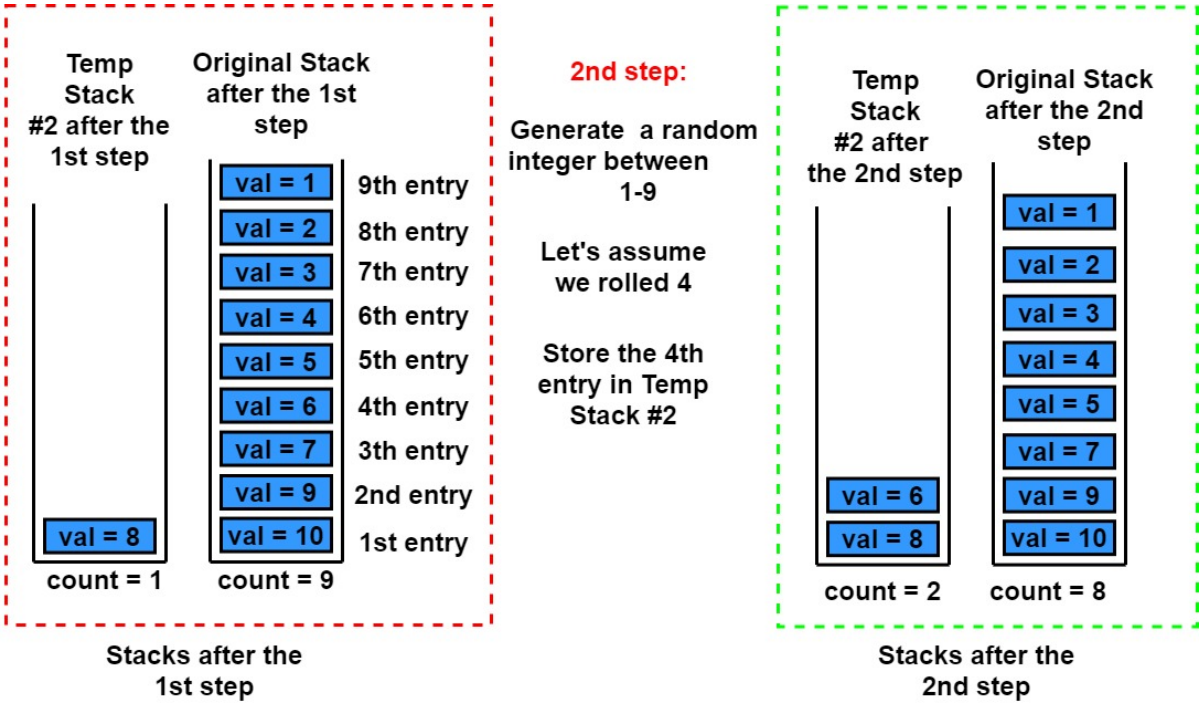


Figure 3: Fisher-Yates shuffle the 2nd step

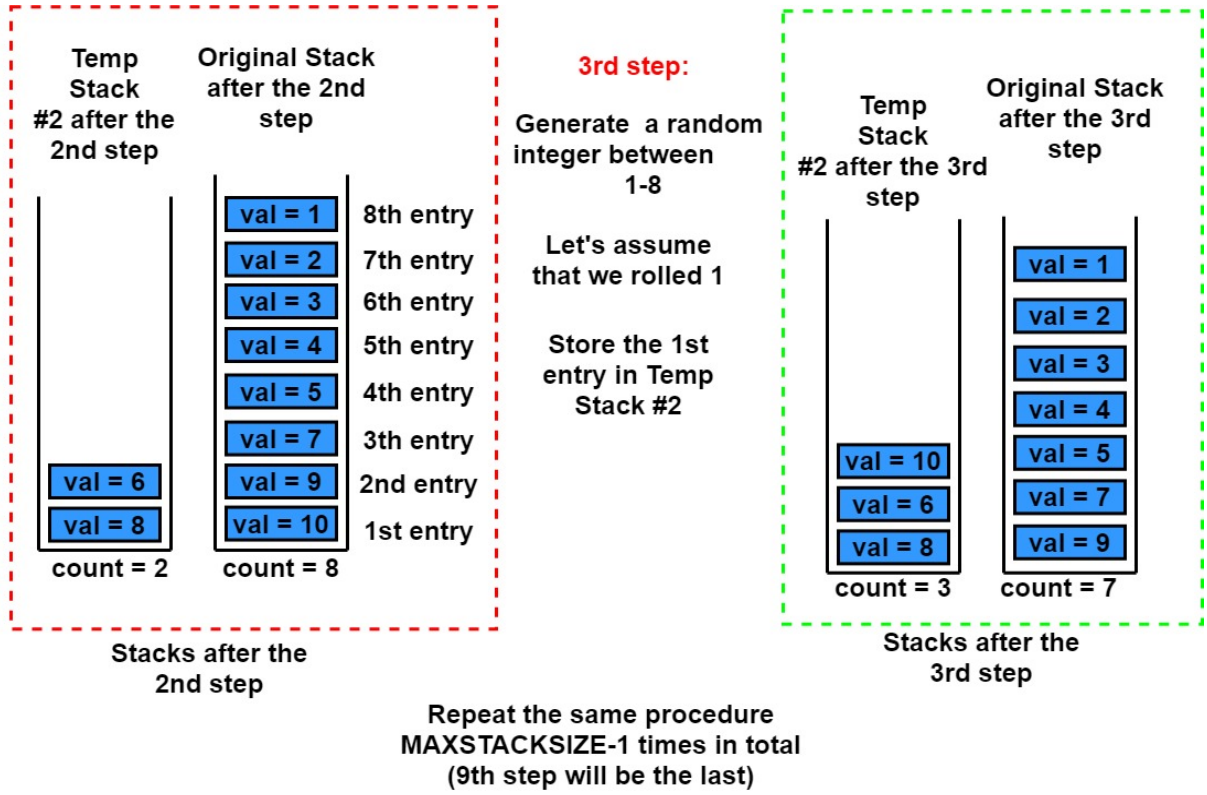


Figure 4: Fisher-Yates shuffle the 3rd step

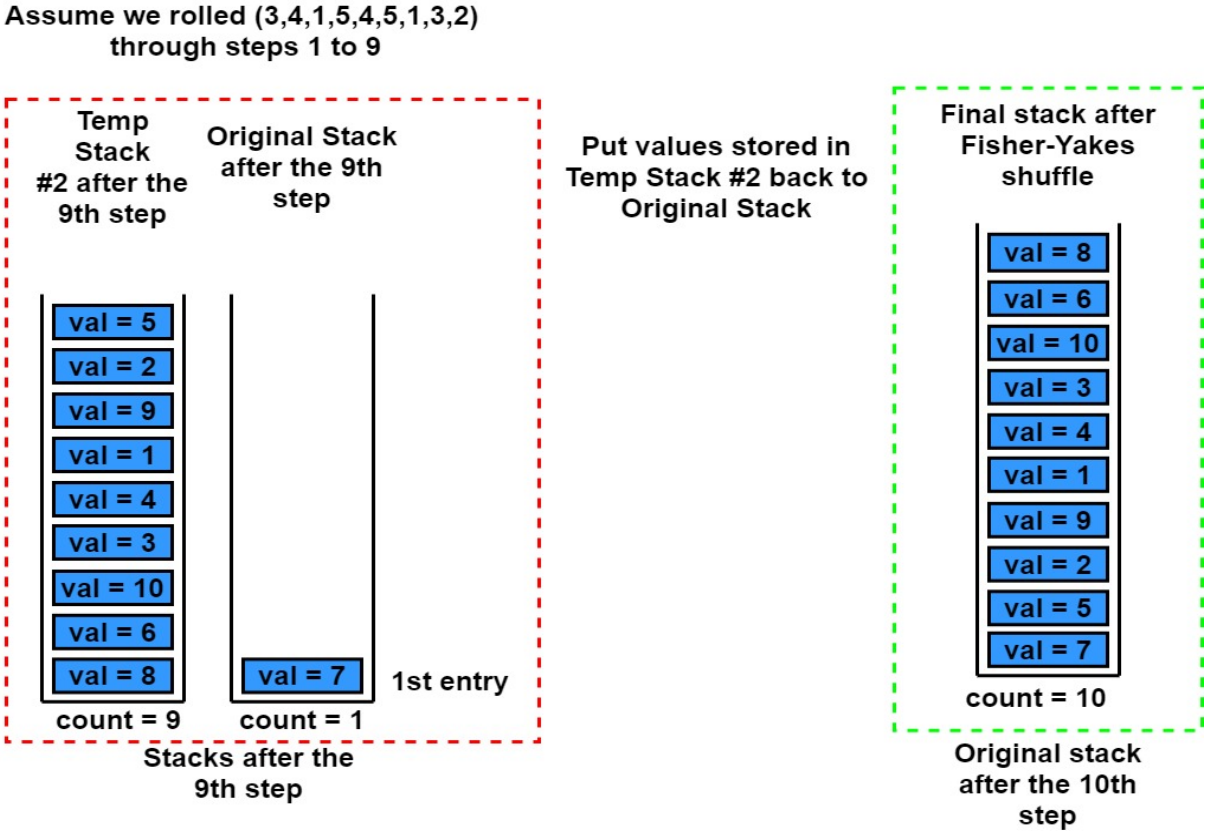


Figure 5: Fisher-Yates shuffle the last step

6- Implement "print" function that prints the values stored in the "Deck" from top to bottom. Follow the steps given below to demonstrate operation of functions you implemented (cf. Figure 6). (15 pts)

```
void print(struct Deck *top)
```

- Declare "Deck" in the main function.
- Fill "Deck" by using "fillDeck" function, where "size" equals MAXSTACKSIZE. Print "Deck" by using "print" function.
- Shuffle "Deck" by using "riffle" function 10 times and print the current state of "Deck" by using "print" function.
- Shuffle "Deck" by using "fisheryates" function and print the current state of "Deck".

```
Original Hand:
1  2  3  4  5  6  7  8  9  10
Riffle Shuffled Hands:
Shuffle 1 : 6  1  7  2  8  3  9  4  10  5
Shuffle 2 : 3  6  9  1  4  7  10  2  5  8
Shuffle 3 : 7  3  10  6  2  9  5  1  8  4
Shuffle 4 : 9  7  5  3  1  10  8  6  4  2
Shuffle 5 : 10  9  8  7  6  5  4  3  2  1
Shuffle 6 : 5  10  4  9  3  8  2  7  1  6
Shuffle 7 : 8  5  2  10  7  4  1  9  6  3
Shuffle 8 : 4  8  1  5  9  2  6  10  3  7
Shuffle 9 : 2  4  6  8  10  1  3  5  7  9
Shuffle 10 : 1  2  3  4  5  6  7  8  9  10

Randomly picked positions: 3  4  1  5  4  5  1  3  2
Shuffled hand (Fisher-yates): 8  6  10  3  4  1  9  2  5  7
```

Figure 6: Screen shot of running program