```
public static void bulgarianSolitaire(int numCards)  {

		//game loop var
		boolean finished = false;

		// Check if given number of cards is triangular
		int n = (int) Math.sqrt(2*numCards);
		if (n*(n+1)/2 != numCards)  {
			System.out.println(numCards + " is not triangular");
			return;
		}

		int shuffle_counter = 1;

		Random rand = new Random();

		//create the final_list
		ArrayList<Integer> final_list = new ArrayList<>();
		int runningSum = 0;
		int next = 1;
		while (runningSum < numCards) {
		   final_list.add(next);
		   runningSum += next;
		   next++;
		}

		//create the original randomized piles
		int remaining_cards = numCards;
		ArrayList<Integer> cards = new ArrayList<Integer>();

		while (remaining_cards > 0) {
			int pile = rand.nextInt(1, remaining_cards+1);
			cards.add(pile);
			remaining_cards -= pile;
		}
		//keep working on it until the final case for the triangular number is
reached
		while (!finished) {
```

```
                    //better logic
                    int cards_removed = 0;

                    for (int i = cards.size()-1; i >= 0; i--) {
                            cards.set(i, cards.get(i)-1);
                            if (cards.get(i) == 0) {
                                    cards.remove(i);
                                    cards_removed++;
                            }
                    }

                    cards.addLast(cards.size() + cards_removed);


                    //output
                    System.out.print("Shuffle #" + shuffle_counter + ": ");
                    for (int i = 0; i < cards.size(); i++) {
                            System.out.print( + cards.get(i) + " ") ;
                    }

                    System.out.println();

                    shuffle_counter++;



                    //checked for size also as containsAll doesn't account for duplicates
                    finished = (cards.size() == final_list.size()
&&cards.containsAll(final_list));

                    // old logic for shuffle, made it more efficient
                                    /*
                                    int original_length = cards.size();
                                    int adaptive_length = original_length;

                                    for (int i = 0; i < adaptive_length; i++) {
                                            cards.set(i, cards.get(i)-1);
                                            if (cards.get(i) == 0) {
                                                    cards.remove(i);
                                                    adaptive_length-=1;
```

```
                                                i--;
                                        }

                                    }

                                    cards.addLast(original_length);
                                    */

            }



        }
}
```