COL 7004 Due: September 7, 2025

Assignment 2

1. Solve the following problems by finding suitable invariants:

   (i) You are given a 64×64 chessboard where a card facing down is kept on each square. We first flip the card on the square $(32, 32)$ (each square is uniquely labelled (i, j) where i and j vary from 1 to 64). Now at each step, we can perform on of the following three operations: (i) flip all the cards in a row, (ii) flip all the cards in a column, (iii) flip all the cards in a 8×8 square of the chessboard. Is is possible to perform a sequence of such operations such that at the end of these operations, all the cards are facing down?

   (ii) Consider the infinite chessboard where each square is labelled (i, j), where i and j can be any integer. You start from the square $(1, 0)$. When you are at a square (i, j), you can move to one of the following squares: $(j, i), (-3i, 2j), (i + 1, j + 4), (i - 4, j - 1), (2i, -3j)$. Can you reach the square $(0, 0)$ through a sequence of such moves?

   (iii) Two players play a game starting with a chocolate consisting of 30×20 square. Starting from the first player, they take turns alternately. In each step, the player can take a piece of a chocolate which is larger than a single square (i.e., has at least 2 rows or columns) and break into two pieces either along the row or along the column (note that initially there is a single piece only). A player loses when there are no such pieces left. Which player has a winning strategy?

2. You are given 3 arrays A, B, C each consisting of a subset of integers arranged in increasing order. You would like to check if there is integer which appears in all three arrays. Consider the following code:

```
i = j = k = 1 # all array indices start from 1
found = false #found will store the desired answer
while (i <= n and j <= n and k <= n): # each array has length n
        if (A[i] == B[j] == C[k]):
             found = true, break
        if (A[i] < B[j]): i++
        else if (B[j] < C[k]): j++
             else k++
```

   What is the loop invariant? Use it to prove correctness of the procedure.

3. Given an array A of integers, you want to compute the index i such that $A[1] + ... + A[i]$ is maximized. Consider the following code:

```
maxsum = -infinity
sum = 0
for i in range(1,n):
    sum = sum + A[i]
    if sum > maxsum: maxsum = sum
```

What is the loop invariant? Use it to argue that *maxsum* stores the desired maximum at the end.

4. You are given a $m \times n$ matrix M where each row and each column is sorted in increasing order. You want to check if x is in M using the following procedure:

```
i = 1, j = n
found = false
while (i <= m and j >= 1):
    if (M[i][j] == x): found = true, break
    else if (M[i][j] > x): j--
    else: i++
```

What is the loop invariant maintained at the start of each iteration? Prove correctness of this algorithm.

5. For each of these counting problems, give a suitable recurrence relation and write the expression for the corresponding generating function:

   (i) The number of binary strings of length n in which any two 1's are separated by at least 2 0's.

   (ii) You walk along a line where each position is labelled $0, 1, 2, ...$ You start at position 0. At each step, you can step forward +1, +2, or +3 steps, but you are not allowed to take two consecutive +2 steps. How many different ways are there of ending at a position n?