CS102 Computer Science II Mid-Term Exam

This exam is closed book. No books, notes, computers, pads and calculators are allowed. All the questions have the same weight. Partial credits will be given. Please write your solution on the exam book. All the programs have to be in Java. There is no need to copy the question. You just need to clearly label your solution with the corresponding question number.

1 Finding a Target in 2D Array with Order

We have an integer 2D array $a$, in which each row and each column have been sorted in the increasing order, i.e., $a[i][j] \geq a[i-1][j]$ and $a[i][j] \geq a[i][j-1]$.

Write a function to find a number in the array. If the number is in the 2D array, return its row and column. It is convenient to wrap the two numbers, row and column, in a Tuple object, so that if the target is not in the array, we can return a null.

(a) A naive implementation would require us to look at each element in the 2D array. The complexity is $O(n \cdot m)$, where $n$ and $m$ are the numbers of the rows and columns of the array. Write the simple implementation.

(b) There is in fact a $O(n + m)$ solution. The idea is to start from the left-bottom corner of the array. Assuming that the array has $N$ rows and $M$ columns, the corner element will be $a[N-1][0]$. We compare the element with the target. If $a[N-1][0] = \text{target}$, we find the number. If $a[N-1][0] < \text{target}$, we can throw away the first column, since all the elements in the first column are not greater than $a[N-1][0]$. Similarly, if $a[N-1][0] > \text{target}$, we can throw away the last row. Now the possible region in $a$ that contains the target number becomes a smaller rectangle. We can take the new left-bottom corner of the region and repeat the previous procedure, until we either find the target or the possible region that has the number becomes empty.

Write a function to implement the fast search method. You can write either a loop version or a recursive version. If you write a loop version, carefully choose the loop invariant and the loop condition.
2 The Sum of Linked Lists

Given two singly linked lists that represent two integers, we would like to compute the summation of the two integers and return the corresponding linked list. For instance, we have list 1 \rightarrow 2 \rightarrow 3 and 8 \rightarrow 9 \rightarrow 5 \rightarrow 7, the summation of the two lists is 9 \rightarrow 1 \rightarrow 9 \rightarrow 7. Note that similarly to grade school summation, there is a carrier forwarded to the next digit, if the local sum is greater than 10.

(a) Use loop to implement the function.

(b) Write a recursive version of the function.

3 Recursion on the Stairs

We climb up an n-step staircase. In each move, we can take one or two steps. Write a recursive function to print out different ways to go through the n-step staircase. Your function should print out all the step combinations as sequences of 1s and 2s. For example, if there is a one-step stair, there is only one way to go upstairs \{1\}. If there is a two-step staircase, there are two ways: two single steps \{1-1\} or one two-step \{2\}.

4 Queue and Stack

Assume that you have a Stack class in Java. Implement a Queue data type with the Stack class. The Stack class has methods: push, peek and pop. The peek method returns the item at the top of the stack without removing the item. pop removes the top item in the stack. Make sure your implementation is as efficient as possible.

(a) Write the Queue class with the stack data type. [Hint: how many stacks do you need to implement the Queue?]

(b) Give a rough estimate about how many operations you need to insert an item into the queue and remove an item from the queue.