CS 4720/5720 Design and Analysis of Algorithms
Homework #3b
Daniel Frey

Answers to homework problems:

1. From book 8.1: *Coin-collecting Problem*

   (a) The algorithm for squares that are inaccessible would differ from the ordinary one by the type of loops. A while loop would be added with bounds for the board and the inaccessible squares. The loop would iterate through the first row/column until an inaccessible square is reached instead of finishing the row. Then it would write the value of an inaccessible square to a large negative number for later comparisons.

   In the for loops for the remaining rows/columns, there would be a check to see if a square is inaccessible, if it is then a large negative number would be written here as well.
   (*Note: -1 denotes inaccessible square*)

   (b) Algorithm: Loop through the first row/column calculating values until inaccessible square is reached. Then loop through the remainder of the row/column setting the value to a large negative number.

   Loop through remaining the rows/columns of board and calculate the number of coins along the way. If a square is accessible, then calculate coins. Otherwise, make value large negative number.

```
CoinCollect(C[1...n, 1...m], B[1...n, 1...m])
    Result[0, 0] = C[0, 0]
    j=1

    //loop through columns of first row
    while j < m and B[0, j] != -1
        Result[0, j] = Result[0, j-1] + C[0, j]
        j = j+1
    for j to m
        Result[0, j] = large negative number

    //loop through remaining columns of each row
    for i = 1 to n
        if B[i, 0] != -1
            Result[i, 0] = Result[i-1, 0] + C[i, 0]
        else
            Result[i, 0] = large negative number
        for j = 1 to m
            if B[i,j] != -1
                Result[i, j] = max(Result[i-1, j], Result[i, j-1]) + C[i, j]
            else
                Result[i, j] = large negative number

    Return Result[n, m]
```

(c) Original grid:

	×		○		
				○	×
×		×	○		○
		○		○	
	○			×	
○				×	

Grid of max coins: *(Note: '–' denotes large negative number.)*

0	–	–	–	–	–
0	0	0	0	1	–
–	0	–	1	1	2
–	0	1	1	2	2
–	1	1	1	–	2
–	1	1	1	–	**2**

Grid of path: *(Note: multiple optimal paths can be taken.)*