CS 1410 Introduction to Computer Science – CS2
Assignment #5: Recursion
Due: 11:59 p.m. Wednesday, February 25, 2015
Total Points: 50 points

You do not need classes to practice recursion. So all the tasks should be solved using recursive functions!

(10 points) In your main function, be sure to use **at least two distinct test cases** to demonstrate each of your recursive functions correctly solves the problem. **For this assignment, you should only submit one .cpp file and two maze files that you used for task 5.**

For debugging purpose, you may add some "cout" statements within the recursive function to show what is happening within each function call. Here are two hints for you:

- **Let the recursion talk to you by printing informative messages.**
- **Write routines to print each data structure, so you can easily see what is happening.**

Task 1: (5 points)

Write a recursive method to determine the sum of the digits of a positive integer. For example, the sum of the digits of 51624 is 5 + 1 + 6 + 2 + 4 = 18. The function prototype is:

```
int ComputeSumOfDigits(int);
```

Task 2: (5 points)

You're standing at the base of a staircase and are heading to the top. A small stride will move up one stair, a large stride advances two. You want to count the number of ways to climb the entire staircase based on different combinations of large and small strides. For example, a staircase of three steps can be climbed in three different ways: via three small strides or one small stride followed by one large stride or one large followed by one small. A staircase of four steps can be climbed in five different ways (enumerating them is an exercise left to you). ☹

Write a recursive function that takes a positive value as the number of stairs and returns the number of different ways to climb a staircase of that height taking strides of one or two stairs at a time. The function prototype is:

```
int CountWays(int) ;
```

Task 3: (7 points)

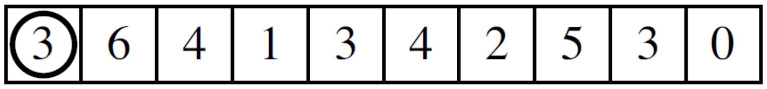
Given a set of integers and a target number, your goal is to find the total number of subsets of those numbers that sum to the given target number. For example, given the numbers {3, 7, 1, 8, -3} and the target sum 4, the subset {3, 1} sums to 4 and the subset {7, -3} sums to 4. So the total number of subsets that sum to 4 is 2. On the other hand, if the target sum is 2, there is no subset that sums to 2. So the total number of subsets that sum to 2 is 0. The prototype for this recursive function is:

```
int CountSubSetSum(int [], int, int) ;
```

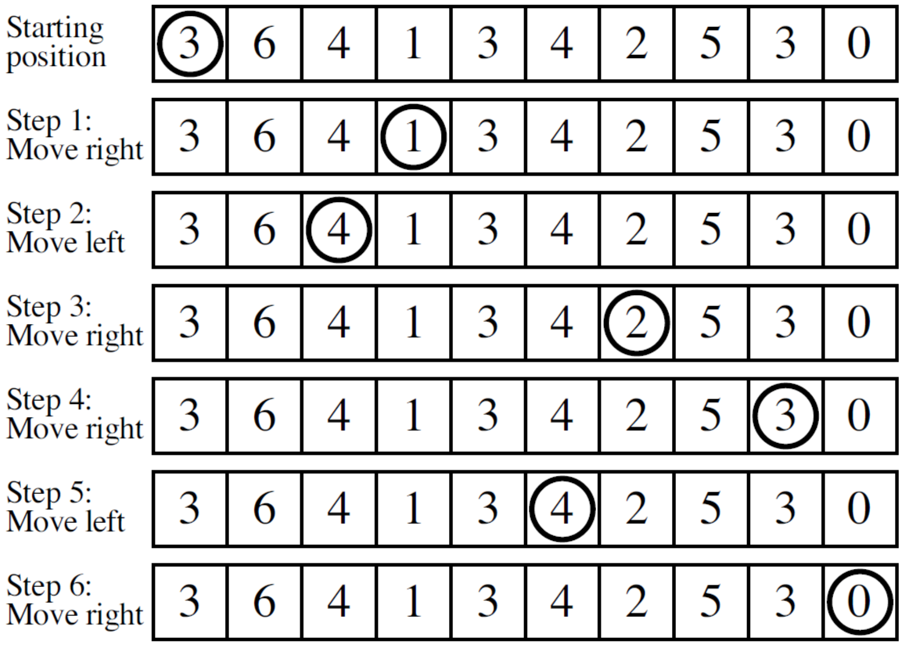
where the first parameter represents an array with several integers, the second parameter represents the number of elements in the array, and the third parameter represents the target number.

Task 4: (8 points)

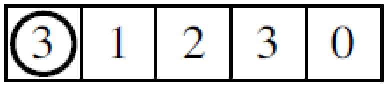
You have been given a puzzle consisting of a row of squares each containing an integer, like this:



The circle on the initial square is a marker that can move to other squares along the row**.** At each step in the puzzle, you may move the marker the number of squares indicated by the integer in the square it currently occupies**.** The marker may move either left or right along the row but may not move past either end**.** For example, the only legal first move is to move the marker three squares to the right because there is no room to move three spaces to the left. The goal of the puzzle is to move the marker to the 0 at the far end of the row**.** In this configuration, you can solve the puzzle by making the following set of moves:



Even though this puzzle is solvable and indeed has more than one solution, some puzzles of this form may be impossible, such as the following one:



In this puzzle, you can bounce between the two 3's, but you cannot reach any other squares.

Write a recursive function that takes a starting position of the marker along with an array containing several integers and the size of the array and returns true if it is possible to solve the puzzle from the starting configuration and false if it is impossible. You may assume that all the integers in the array are positive except for the last entry, the goal square, which is always zero. The values of the elements in the array must be the same after calling your function. The function prototype is:

bool IsSolvable(int, int [], int) ;

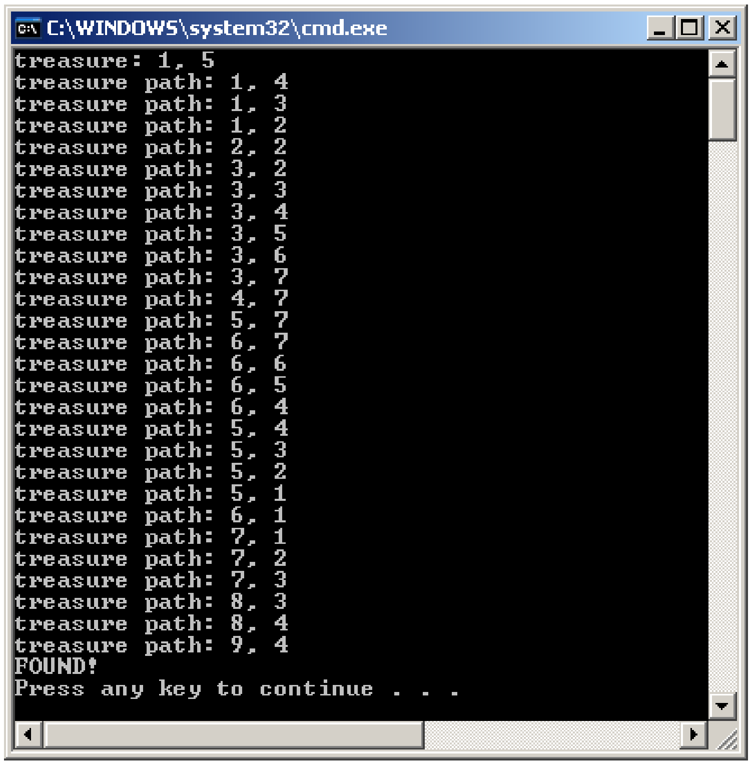
where the first parameter represents the starting position of the marker, the second parameter represents the array with several integers, and the third parameter represents the number of elements in the array.

Task 5: (15 points)

Solve a 2-D maze read from a maze file (**maze.txt**), which consists of three segments: The first line contains the number of rows and columns in the maze. The second line contains the starting position of the explorer. The rest of the file contains the maze – one row per line. Walls are represented by 'X', paths are represented by '.', and the treasure is represented by a 't'. Specifically, your recursive function should output a valid path between the treasure and the explorer. You may study the materials at the following URL: https://www.cs.bu.edu/teaching/alg/maze/ to learn how to solve the problem. For example, here is an 11 × 10 maze with the explorer starting at the cell (9,4). Note: The most upper left cell location is (0, 0).

```
11 10
9 4
XXXXXXXXXX
XX...tX..X
XX.XXXX.XX
XX.......X
XXXXXXX.XX
X....XX.XX
X.XX.....X
X...XX.XXX
XXX..XXXXX
XXXX....XX
XXXXXXXXXX
```

The output of your program should look like this:


C:\WINDOWS\system32\cmd.exe

```
treasure: 1, 5
treasure path: 1, 4
treasure path: 1, 3
treasure path: 1, 2
treasure path: 2, 2
treasure path: 3, 2
treasure path: 3, 3
treasure path: 3, 4
treasure path: 3, 5
treasure path: 3, 6
treasure path: 3, 7
treasure path: 4, 7
treasure path: 5, 7
treasure path: 6, 7
treasure path: 6, 6
treasure path: 6, 5
treasure path: 6, 4
treasure path: 5, 4
treasure path: 5, 3
treasure path: 5, 2
treasure path: 5, 1
treasure path: 6, 1
treasure path: 7, 1
treasure path: 7, 2
treasure path: 7, 3
treasure path: 8, 3
treasure path: 8, 4
treasure path: 9, 4
FOUND!
Press any key to continue . . .
```