

CS 1410 Introduction to Computer Science – CS2
Section 1: MWF 9:30 a.m. – 10:20 a.m.
Instructor: Xiaojun Qi
Assignment #9: Recursion

Given: Sunday, March 24, 2013
Due: 11:59 p.m. Monday, April 1, 2013
Total Points: 35 points

In order to practice writing recursive functions, **you do not need to use classes for this assignment. All the following four tasks should be solved using recursion!** In your main function, be sure to demonstrate your recursive functions with enough test cases to prove to the grader that you have solved the problem correctly (two distinct cases should be enough for each problem). **For this assignment, you should only submit one .cpp file!**

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)

Suppose that you have been transported back to 1777 and the Revolutionary War. You have been assigned to evaluate the amount of ammunition available to the British for use with their large cannon. Fortunately, the British have stacked the cannonballs into a single pyramid-shaped stack with one cannonball at the top, sitting on top of a square composed of four cannonballs, sitting on top of a square composed of nine cannonballs, and so forth. However, you only have time to count the number of layers in the pyramid before you are able to escape back to your own troops. Your mission is to write a recursive method `cannonball` that takes the height of the pyramid as its argument and returns the number of cannonballs therein.

```
int Cannonball(int levels)
```

Task 2: (5 points)

Write a function to recursively print the binary equivalent of a positive integer *N*. An example of the binary equivalent of 13 may be found by repeatedly dividing 13 by 2 until the quotient is 0. If the remainder is 1, 1 gets printed. Otherwise a zero gets printed.

```
13 / 2 = 6 remainder 1
6 / 2 = 3 remainder 0
3 / 2 = 1 remainder 1
1 / 2 = 0 remainder 1
```

13 in base 2 is 1101 (reverse the remainders obtained in the division process).

```
void PrintBinary(int n)
```

Task 3: (5 points)

Write a recursive method to determine the sum of the digits of an integer. For example: the sum of the digits of 51624 is $5 + 1 + 6 + 2 + 4 = 18$

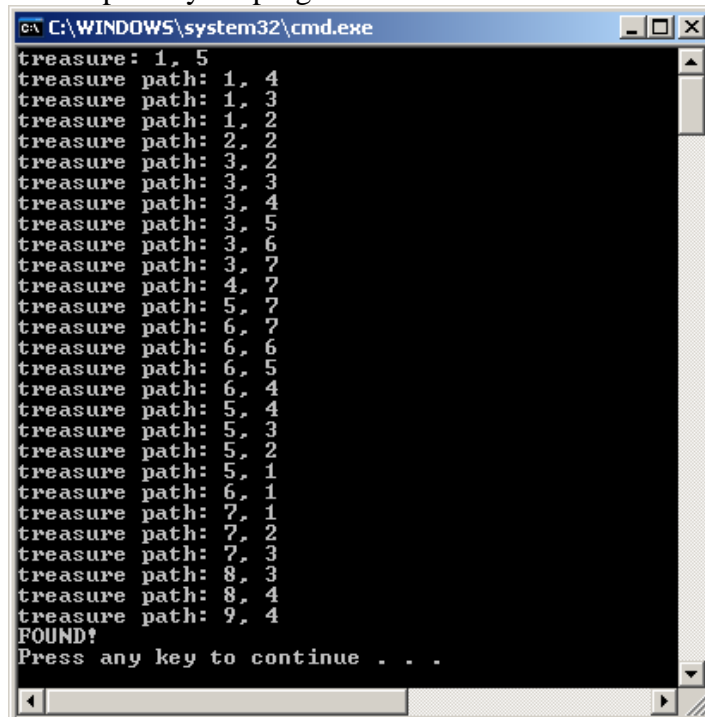
```
int SumOfDigits(int N)
```

Task 4: (20 points)

Write a C++ program to solve a 2-D maze read from a disk file. A maze file (**maze.txt**) 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. For example, here is an 11 x 10 maze with the explorer starting in cell (9,4).

```
11 10
9 4
XXXXXXXXXX
XX...tX..X
XX.XXXX.XX
XX.....X
XXXXXXXXXX
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 . . .
```