Homework 5

Recursion

CS 5060 Intensive Programming, Fall 2012

100 points

Due: 3:59 pm October 15, 2011

Solve the following problems (100 points).

There are easier and harder problems. Harder problems are worth more points. You must use recursion to solve all the problems. You will get a *zero score* for any problem that you solve *without recursion* (even if the result is correct).

Note: All input must be read from the standard input stream, and all output must be written to the standard output stream. For this assignment, assume the input is correct.

Hint: *Start with the easy problems and start early.*

Submission.

Submit a `zip` file with the following files:

1. Eight code files `Count.java`, `SumDigits.java`, `RBinarySearch.java`, `ArithmeticExpression.java`, `SubgroupSum.java`, `SubgroupSplitSum.java`, `ConnectedComponents.java`, `MapReachability.java` with the solutions to problems 1, 2, 3, ..., 8, respectively.

Include your name and A number at the top of each source file. Name the `zip` file `hw05_firstName_lastName.zip`. For example, if your name is John Smith, name the file `hw05_John_Smith.zip`.

Problem 1: Count (easy, 5 points)

Input: The input begins with the number t of test cases in a single line ($t \leq 100$). Each test case has two lines. The first line has two numbers: n ($1 \leq n \leq 1000$) and x ($1 \leq x \leq 1000000$). The second line has a list of n numbers m ($1 \leq m \leq 1000000$) separated by spaces.

Output: For each test case output the number of occurrences of x in the list.

Example:

Input:

```
3
4 0
5 0 6 0
4 3
5 0 6 0
7 17
2 3 5 7 11 13 17
```

Output:

```
2
0
1
```

Problem 2: Sum digits (easy, 5 points)

Input: The input begins with the number t of test cases in a single line ($t \leq 100$). Each of the next t lines contains a number n ($0 \leq n \leq 1 \times 10^{18}$).

Output: For each test case? output the sum of the digits of n.

Example:

Input:

```
3
123
5060
72057594037927936
```

Output:

```
6
11
85
```

Hint: *Use division and modulo.*

Problem 3: Recursive binary search (medium, 10 points)

Input: The input begins with the number t of test cases in a single line ($t \leq 100$). Each test case has two lines. The first line has two numbers: n ($1 \leq n \leq 1000$) and x ($1 \leq x \leq 1000000$). The second line has a list of n numbers m ($1 \leq m \leq 1000000$) sorted in increasing order and separated by spaces.

Output: For each test case determine if the number x is in the list.

Example:

Input:

```
3
4 0
0 0 5 6
4 3
0 0 5 6
7 17
2 3 5 7 11 13 17
```

Output:

```
YES
NO
YES
```

Problem 4: Arithmetic expressions (hard, 20 points)

Implement a recursive function to multiply two integers and a recursive function to divide two integers. You can use the addition (+) and subtraction (−) operators, but you cannot use the multiplication (⋆) or division (/) operators of the language in your program. Use these functions to evaluate arithmetic expressions (you are not required to use recursion for the evaluation part of this problem). *Remember that multiplication and division have preference over addition and subtraction.*

Input: The input begins with the number t of test cases in a single line ($t \leq 100$). Each of the next t lines contains an arithmetic expression. The expression only has integer numbers and it can have parenthesis and any of the following binary operators: +, −, ⋆, /. The division is integer division (there is no division by zero).

Output: For each test case output the result of evaluating the expression (the result could be negative).

Example:

Input:

```
4
(7 - 2) * 3
7 * 2 - 3
(3 - 8) / 2
3 - 8 / 2
```

Output:

```
15
11
-2
-1
```

Hint: *The implementations explained in class using only the increment (++) and decrement (−−) operators will be two slow.*

Problem 5: Subgroup sum (hard, 15 points)

Input: The input begins with the number t of test cases in a single line ($t \leq 100$). Each test case has two lines. The first line has two numbers: n ($1 \leq n \leq 20$) and x ($1 \leq x \leq 1000000$). The second line has a list of n numbers m ($1 \leq m \leq 1000000$) separated by spaces.

Output: For each test case determine if there is a subgroup S of the numbers in the list, such that the numbers in S add up to x.

Example:

Input:

```
3
4 3
5 0 6 0
4 6
3 2 1 4
7 23
2 3 5 7 11 13 17
```

Output:

```
NO
YES
YES
```

Hint: *Note that numbers in the subgroup do not have to be consecutive.*

Problem 6: Subgroup split sum (hard, 15 points)

Input: The input begins with the number t of test cases in a single line ($t \leq 100$). Each test case has two lines. The first line a number: n ($1 \leq n \leq 20$). The second line has a list of n numbers m ($1 \leq m \leq 1000000$) separated by spaces.

Output: For each test case determine if list can be divided into two non-empty subgroups such that the sum of each subgroup is the same.

Example:

Input:

```
3
4
5 0 6 0
4
3 2 1 4
7
2 3 5 7 11 13 17
```

Output:

```
NO
YES
YES
```

Hint: *Note that numbers in the subgroups do not have to be consecutive, so this is not the same as the array split sum problem we saw before.*

Problem 7: Connected components (hard, 15 points)

Read about pixel connectivity at `http://en.wikipedia.org/wiki/Pixel_connectivity`

Input: The input begins with the number t of test cases in a single line ($t \leq 100$). Each test case has multiple lines. The first line has two numbers: a number of rows r ($1 \leq r \leq 1000$) and a number of columns c ($1 \leq c \leq 1000$). The next r lines describe a binary image.

Output: For each test case output the number of 4-connected components in the image.

Example:

Input:

```
2
4 5
11100
01100
01111
00001
4 5
11100
01100
01111
10001
```

Output:

```
3
5
```

Problem 8: Map reachability (medium-hard, 15 points)

Input: The input begins with the number t of test cases in a single line ($t \leq 100$). Each test case has multiple lines. The first line has two numbers: a number of rows r ($1 \leq r \leq 1000$) and a number of columns c ($1 \leq c \leq 1000$). The next r lines describe a map where `'.'` are empty spaces, `'X'` are walls, `'S'` is your starting location, and `'G'` is your goal destination.

Output: For each test case determine if it is possible to reach the goal from your starting location (use 4-connectivity).

Example:

Input:

```
2
4 5
XXX.G
.XX..
.XXXX
S...X
4 5
XXX.G
.XX..
.XX.X
S...X
```

Output:

```
NO
YES
```