Informatics II, Spring 2024, Exercise 2

Publication of exercise: February 26, 2024
Publication of solution: March 4, 2024
Exercise classes: March 4 - March 8, 2024

Learning Goal

- Understand how to define the subproblem of a recursive problem.
- Learn how to define the stopping conditions of a recursive program.

Task 1: Palindrome [Easy]

A **palindrome** is a word, number, phrase, or other sequence of symbols that reads the same backwards as forwards, such as 'madam' or 'racecar'.

Write an **recursive** algorithm to check whether a given string is a palindrome. The string can include numbers, letters (case-sensitive), and other symbols. For example, **'1a_b3cD45t54Dc3b_a1'** is a palindrome.

You can start by filling out the function below.

```c
/**
* Check the symbols in index i and index j
*/
int isPalindrome(char X[], int i, int j) {
    //Put your code here
}

int main(){
    char X[] = "1a_b3cD45t54Dc3b_a1";
    //Put your code here
}
```

Task 2: Eight Queen Puzzle [Medium]

The **Eight Queens Puzzle** is the problem of placing eight chess queens on an 8×8 chessboard so that no two queens threaten each other. Therefore, a solution must satisfy that no two queens share the same row, column, or diagonal. One possible solution is shown in Figure 1.

The eight queens puzzle is a special case of the more general **N Queens Problem** of placing N non-attacking queens on an $N \times N$ chessboard. In this task, you are required to calculate the number of all possible solutions for the N queens problem. Solutions exist for all natural numbers N with the exception of $N = 2$ and $N = 3$.

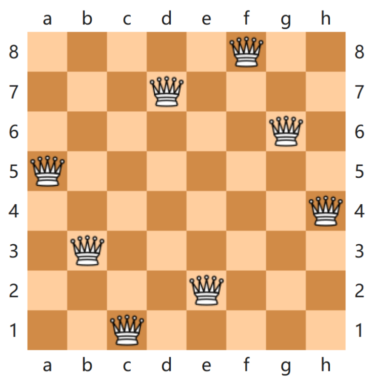

Figure 1: One solution for the eight queen puzzle

Given the number of queens N ($3 < N < 20$), you need to print the number of all possible solutions. You can start by filling out the functions below.

```
int N = 8;
int count = 0;

/**
* Check if it's safe to place a new Queen in the given position
*/
int isSafe(int board[N][N], int row, int col) {
    //Put your code here
}
/**
* Try to place the N queens in the board
* and update the global variable 'count'
*/
void solveNQueens(int board[N][N], int row) {
    //Put your code here
}

int main() {
    int board[N][N];
    memset(board, 0, sizeof(board)); //initialize the board and fill with 0
    //Put your code here
}
```

Task 3: Tower of Hanoi [Medium]

The **Tower of Hanoi** is a game consisting of three columns and a number of disks that can be placed on any of the columns. No two disks have the same size. At the beginning, all disks are stacked on the first column (the left one) in order of decreasing size (the smallest disk at the top). The goal of the game is to move all disks to the second column (the middle one). There are only two rules:

- In each step only the top-most disk from a column can be moved to the top of another column.
- A large disk may never be placed on top of a smaller disk.

Figure 2 illustrates the stacks after 5 moves in a game with 6 disks .

Figure 2: Tower of Hanoi with 6 disks

Your task is to display all steps in a game and count the number of steps you play. You can show them in formats like **'step 13: a ->b'** or **'step 13: T1 ->T2'**.

You can start by filling out the function below. You should do it in a recursive way .

```
int moveCount = 0;
/**
* Move n disks from source to destination via auxiliary
*/
void hanoi(int n, char src, char aux, char dst) {
    //Put your code here
}

int main() {
    //Put your code here
}
```

Task 4: Calculator [Hard]

In this task, you can implement your own calculator! Suppose you are given a numerical expression with only basic operators $(+, -, \times, \div)$ and parentheses $()$.

You can assume that:

- There is no blank inside the string.
- All numbers are integers, and negative integers are always enclosed in parentheses.

Please calculate the result of a numerical expression using a recursive function and print the result. Below are the example inputs and outputs.

Inputs & Outputs

```
(3+4)*5/6+7*8*(1+2-3)
Ans = 5.833333

(3+4)*5/6+7*8*(1+(-2)-1+1)
Ans = -50.166668
```

You can start by filling out the functions below.

```c
/**
* Find the position of the operator with the lowest
* priority between index h and index t
*/
int findOperator(char *expr, int h, int t) {
    //Put your code here
}
/**
* Calculate the result of the expression between index h and index t
*/
float calculator(char *expr, int h, int t) {
    //Put your code here
}

int main() {
    char expr[] = "(3+4)*5/6+7*8*(1+2-3)";
    //Put your code here
}
```