Assignment 1

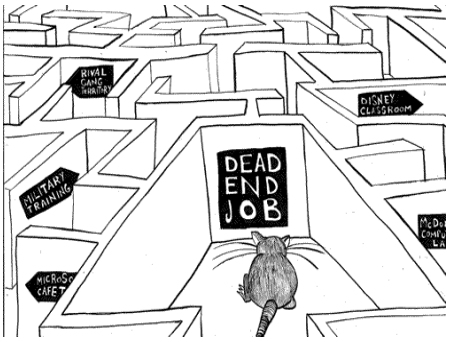
Issue Date: Saturday – May 21th, 2022
Submission Deadline: Friday – May 27th, 2022 (Till 5:00 pm) **Marks = 60**

Instructions!

1. You are required to do this assignment on your oswn. Absolutely **NO** collaboration is allowed, if you face any difficulty feel free to discuss with me.
2. Cheating will result in a **ZERO** for the assignment. (Finding solutions online is cheating, Copying someone else's solution is cheating). Also, do not hand your work over to another student to read/copy. If you allow anyone to copy your work, in part or in whole, you are liable as well.
3. You programs should be running properly.
4. Hard **DEADLINE** of this assignment is **Friday, May 27th, 2022**. No late submissions will be accepted after due date and time so manage emergencies beforehand.

Task 01: [30 Marks]

"Rat in a Maze" is a famous backtracking problem, where a rat starts from a source and has to reach the destination. In the maze, some cells are valid to move and some cells are blocked. At any point, the rat can move one of four directions i.e. forward, backward, left and right. The task is to check if there exists any path so that the rat can reach the destination or not, if it exists then mark the path for the rat.



You are required to implement a simple version of this problem in which you have to find the success path from given source to destination. A relatively complex version can be to find the success path with a limited number of moves. Let's stick to the simpler version. The maze will be represented as form of a two dimensional binary matrix. Where,

1. 0's represent the open path whereas 1's represents barriers/brick walls.
2. The legal moves are left, right, top and down from a given cell (Obviously, you are not allowed to move off the maze).
3. You are required to read maze from a file and stores your output to another file.

Input:

A file named **in.txt** will contain the input matrix which represents the Maze. First line of the file shows the order of the matrix. Second and third lines show the source and destination respectively. The maze matrix is given from fourth line onwards.

Output:

The output of your program should be stored in a text file **out.txt**. The first, second and third line should be same as input file. At fourth line the success path should be printed and after that the maze matrix should be printed such that the success path should be shown with asterisk.

Sample Input File Format:

```
8
0 0
7 7
0 0 0 1 1 0 0 0
0 1 0 0 1 0 0 0
0 0 1 0 0 0 1 0
0 0 1 1 1 1 1 0
0 0 0 0 1 0 0 0
0 0 0 0 1 1 1 0
0 1 1 0 0 1 0 0
0 1 1 1 0 1 0 0
```

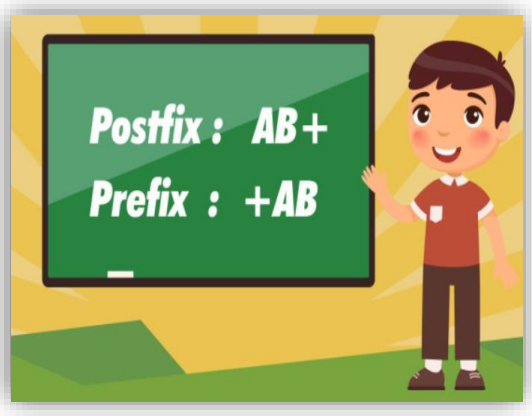
Sample Output File Format:

```
8
0 0
7 7
(0,0) (0,1) (0,2) (1,2) (1,3) (2,3) (2,4) (2,5) (1,5) (1,6) (1,7) (2,7) (3,7) (4,7) (5,7) (6,7) (7,7)

* * * 1 1 0 0 0
0 1 * * 1 * * *
0 0 1 * * * 1 *
0 0 1 1 1 1 1 *
0 0 0 0 1 0 0 *
0 0 0 0 1 1 1 *
0 1 1 0 0 1 0 *
0 1 1 1 0 1 0 *
```

Task 02: [20 Marks]

A mathematical expression is made up of operands, operators and delimiters. We have several ways of writing mathematical expressions. Infix, postfix and prefix notations are three different but equivalent ways of writing expressions.



Infix notation is the usual way we write expressions, in which operators are written in between the operands.

$$A \times (B + C)/D$$

Infix notation needs extra information to make the order of evaluation of the operators clear, for example rules built into the language about operator precedence and associativity, and brackets ( ) to allow users to override these rules. For example, the usual rules for associativity say that we perform operations from left to right, so the multiplication by A is assumed to come before the division by D. Similarly, the usual rules for precedence say that we perform multiplication and division before we perform addition and subtraction.

Postfix notation (also known as "Reverse Polish Notation") is the way of writing expressions in which operators are written after their operands. The order of evaluation of operators is always left-to-right, and brackets cannot be used to change this order. The infix expression given above is equivalent to

$$ABC +\times D/$$

In **Prefix notation** (also known as "Polish notation") operators are written before their operands. Operators are evaluated left-to-right and brackets are superfluous. Operators act on the two nearest values on the right. The expressions given above are equivalent to

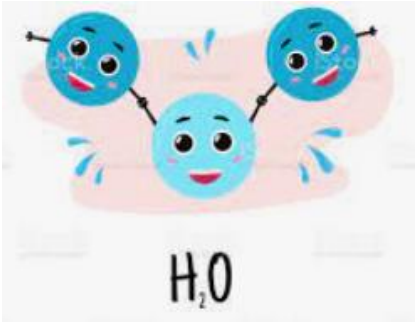
$$/\times A + BCD$$

We have already discussed Infix to postfix conversion in the class. Now, you are required to implement this conversion. After converting the expressions also evaluate the expression and return result. Implement following three functions:

1. **isBalanced (expression):** This function should check whether the given infix expression is correct and balanced or not. We will consider an expression correct if the parenthesis are correctly added and are balanced. For example: $(a \times \{b + c\} - 4/x\ + [e - 5])$ is a correct expressions. While, $(5 + \{6 \times) - 2\}$ and $\{5 + z)$ are examples of incorrect expressions.
2. **infixToPostfix (expression):** This function should take an infix expression as input and returns equivalent postfix expression.
3. **Evaluate (expression, key):** This function should take an infix expression or a postfix expression as first argument and takes a Boolean key as second argument (0 for infix expression, 1 for postfix expression) and returns the value of the expression after evaluation.

Also write a proper main program providing menu to make it easy to test your functions. No marks shall be given without this driver program.

Task 03: [10 Marks]



Write a program to take a chemical formula (given as a string), and return the count of each atom. An atomic element always starts with an uppercase character, then zero or more lowercase letters, representing the name. One or more digits representing the count of that element may follow if the count is greater than 1. If the count is 1, no digits will follow. For example, H2O and H2O2 are possible, but H1O2 is impossible. Two formulas concatenated together produce another formula. For example, H2O2He3Mg4 is also a formula. A formula placed in parentheses, and a count (optionally added) is also a formula. For example, (H2O2) and (H2O2)3 are formulas.

Given a formula, output the count of all elements as a string in the following form: the first name (in sorted order), followed by its count (if that count is more than 1), followed by the second name (in sorted order), followed by its count (if that count is more than 1), and so on. For example,

Formula = K4(ON(SO3)2)2

Output: K4N2O14S4

How to submit your Assignment

1. Name your projects as task1, task2 and task3. Put all your projects in a folder named rollno-Assignment 1-Your Name. for example: BCSF19A32-Assignment 1-Mohsin Raza.
2. Compress the folder and paste it into your class folder on teacher's data appropriately. For example, follow the path \\printsrv\Teachers Data\madiha\ and place your assignment in BCSF19A-Assignment1 folder.
3. Remember this folder will disappear after due Date and time so be careful in placing your assignments on time.
4. Feel free to contact me if you face any difficulty in submission process.

Good Luck!