2024_DS_Fall_Homework 1

Notice

The deadline is 2024/10/29 23:59. Homework should be submitted as a c source file, not an executable file. In your homework assignment, read the input from stdin and write your output to stdout. The file's name should be hw1_p1.c.

Problem 1: Prefix, Infix, and Postfix Expression Conversion (2%)

Expressions in prefix, infix, and postfix notations are commonly used in computer science for parsing and evaluating arithmetic operations. In this problem, you will implement functions to transform these expressions between their respective forms while maintaining the correct operator precedence.

Notes:

1. The test data doesn't contain parentheses or space characters.
2. The operators can only be +, -, *, and /.

(a) Prefix to Infix Conversion

Write a C function that converts a **prefix expression** into an **infix expression**. A prefix expression places the operator before the operands (e.g., `+ab`) . In contrast, an infix expression places operators between operands (e.g., `a+b`). Your function should process the given prefix expression and convert it to its infix equivalent.

Example:

Sample Input:

```
/+*abcd
```

Sample Output:

```
a*b+c/d
```

(b) Infix to Postfix Conversion

Write a C function that converts an **infix expression** into a **postfix expression** . In an infix expression, operators are placed between operands (e.g., `a+b`). In postfix notation, operators follow their operands (e.g., `ab+`)

Example:

Sample Input:

```
a*b/c
```

Sample Output:

```
ab*c/
```

Problem 2: Priority Queue Implementation Using Min-Heap (2%)

In this problem, you are required to implement a **Min-Priority Queue** using a **Min-Heap**, where the element with the smallest value is considered to have the highest priority.

You need to implement the insertion and deletion operations on the Min-Heap and output the final state of the heap in **level-order** after performing all the operations specified in the input.

Input Format:

- The input consists of a series of commands. Each command will either be an insertion of a number into the heap (e.g., `insert 3`) or a deletion of a number from the heap (e.g., `delete 3`).
- The commands will be executed sequentially.

Output Format:

- After all operations are performed, output the elements of the Min-Heap in **level-order**, with **each number followed by a space character.**

Notes:

1. To ensure the uniqueness of the output, follow the **Max-Heap Operation** steps as described in **Chapter 5.6** of the textbook, but modify them for a **Min-Heap**. This ensures consistency in how elements are inserted and deleted.
2. Each number in both the input and output should be followed by a space character.

3. The final heap should be represented using **level-order traversal**, where nodes are printed from top to bottom, left to right.

Example:

Sample Input:

```
insert 3
insert 30
insert 15
insert 4
insert 7
insert 18
delete 3
delete 7
```

Sample Output:

```
4 18 15 30
```

Problem 3: Breadth-First Search (BFS) Implementation on an Undirected Graph (3%)

In this problem, given an undirected graph G with m vertices that is represented by an $m \times m$ adjacency matrix **A**, you will implement BFS on G. The goal is to output the order of vertices visited during the BFS traversal.

Input Format:

- The first line contains an integer m, representing the number of vertices in the graph. The vertices are numbered from 1 to m, i.e., the vertex set is $V=\{1,2,3,\ldots,m\}$.
- The following $m \times m$ matrix represents the adjacency matrix **A** of the undirected graph G, where each element in the matrix is either `0` or `1`.
    - **A**[i][j]=1 indicates that there is an edge between vertex i and vertex j.
    - **A**[i][j]=0 indicates no edge between vertex i and vertex j.

Output Format:

- Output a single line containing the BFS traversal order, starting from vertex 1. If there are multiple vertices that can be visited at the same step, the vertex with the smallest index should always be visited first.

Notes:

1. The BFS should continue until all vertices reachable from the starting vertex have been visited.
2. There are no self-loops or multiple edges in *G*.
3. Ensure that both input and output contain a space character after each number.

Example:

Sample Input:

```
4
0 1 0 1
1 0 1 1
0 1 0 1
1 1 1 0
```

Sample Output:

```
1 2 4 3
```