**STACK & QUEUE (20%)**

1. What are the main difference between Stack and Queue? (.pdf)
2. Explain prefix, infix, and postfix notation and its implementation using stack!

Answer:

1. Main difference between Stack and Queue

|  |  |  |
| --- | --- | --- |
| No. | Stack | Queue |
| 1. | Data stored to Stack in Last In First Out (LIFO) method. As if we want to get a plate in and out of a pile of plates, we always add and remove elements from the topmost element. | Data stored in First In First Out (FIFO) method. As if we’re waiting in a queue, the person that comes in first will also come out first. |
| 2. | Thus, we need Push Head and Pop Head function for Stack. | Thus, we need Push Tail and Pop Head function for Queue. |

1. Prefix, Infix, Postfix Notation and Its Implementation using Stack
2. **Prefix**: operator is written before two operands.
3. **Postfix**: operator is written after two operands.
4. **Infix**: operator is written between two operands.

**Note**:  
We should always consider the highest predence in infix operation.

**Ex**:

1. **Infix Notation** – {(4 + 2) \* 3 – 10}. () is the highest predence among the operands, therefore we add 4 with 2 first, which will become 6. Now, the operation left becomes 6 \* 3 – 10. In this current operation, the highest precedence is \*, thus we multiply 6 with 3 = 18. Afterwards, the operation left becomes 18 – 10. In this current result, we can see that the highest precedence is -. Hence, we minus 18 with 10, which becomes 8 as the final result.
2. **Prefix Notation** – (/ ↑ + 3 2 \* 1 3 5). In Prefix operation, we’ll consider from the backmost operator and operands first. The two backmost operands with the backmost operator (\*) before them are 1 and 3. Therefore, we calculate 1 \* 3 = 3, which will result (/ ↑ + 3 2 3 5). In this current operation, we can see that two operands before the backmost operator (+) are 3 and 2. Thus, we add 3 with 2 = 5 and therefore the operation becomes / ↑ 5 3 5. We keep continuing the previous processes till there’s no operator and operand left. If we repeat the previous processes, then we will consider ↑ 5 3 since it’s the backmost operator with two operands before. ↑ means power, so 5 power 3 becomes 125. Now, it’s only / 125 5 left, which its calculation (5 divides 125) will result 25 as the final result.
3. **Postfix Notation** – (100 10 / 15 + 18 9 / ↑). Since the frontmost operator is / with the two operands before, therefore we calculate 100 10 / first, which is 100 / 10 = 10. The operation therefore will be 10 15 + 18 9 / ↑. In this current operation, the frontmost operator is + with two operands: 10 and 15 before. We’ll repeat the previous process with adding 10 with 15, which results 25. Now, the operation left becomes 25 18 9 / ↑. In this current operation, the frontmost operator with two operands before is / with 18 and 9 as the two operands. Thus, we calculate 18 / 9 = 2. After that, the operation left becomes 25 2 ↑, which will result 25^(2) = 625.



