**Question 1:** **(5 Marks)** a. Write a C++ program to convert an **infix** expression that includes (, ), +, -, *, and / to **postfix**. **Hint**: use **stacks** to implement this conversion.

b. Write a C++ program to convert a **postfix** expression to **infix**. **Hint**: Use **binary tree** algorithms to implement this conversion.

## Question 2: (15 Marks)

A queue is a sequence of elements, all of the same type, to which elements can be appended at one end (the rear of the queue) and from which elements can be removed at the other end (the front). Queues are first-in-first-out (FIFO) structures that mimic the behavior of such systems as people in lines, cars at traffic lights, and files to be printed.

Systems that involve queues are called **queuing systems**. In general, a queuing system consists of one or more queues of elements waiting to be served by one or more servers. When an element is removed from the front of a queue, a server serves that element. How queues and servers interact and parameters such as the numbers of queues and servers, how often new elements arrive, and how often servers remove elements from queues determine the behavior of a queuing system.

A **queuing simulation** is a program that simulates a queuing system. A probabilistic simulation calls a pseudo-random number generator to determine if events occur at each tick of the simulation's clock.

### Description

You are required to design, implement, test and document a probabilistic C++ simulation of a queuing system with several queues and several servers, one server for each queue, as in a grocery or discount store. Because there are several lines, each arriving customer always joins the line that is currently the shortest. If several lines are equally short, the new arrival may join any one of them. A line with a free teller is "shorter" than one whose teller is occupied. Once in a line, a customer does not leave it until served.

The program will read from the terminal the parameters of the simulation---how many queue/server pairs (maximum 10), the longest time for a customer transaction, the probability that a customer arrives during a single tick, the duration of the simulation, and a seed value for the random number generator.

It will simulate the queuing system for the specified duration, showing a picture of the system at every clock tick, then report some statistics: the average time each customer waited, the longest time any customer waited, the number of customers served, and the numbers of customers left in queues.
The number of queue/server pairs, of course, is at least 1. The probability that a customer arrives during one tick is a percentage, thus an integer between 0 and 100. Each transaction's time is random, uniformly distributed between one tick and a positive maximum value entered from the terminal.

### INPUT

From the terminal the program will read:

- the number of queue/server pairs up to 10 (lines and their tellers),
- the probability that a customer will arrive in a single tick (a percentage, thus an integer between 1 and 100),
- the longest time a transaction may take, a positive integer,
- the duration of the simulation in ticks, at least 1, and

- an integer seed value for the pseudo-random number generator.

## OUTPUT

The program's output, directed to the terminal, will be a snapshot of the system at each clock tick during the simulation, followed by these statistics:

- the number of customers served,
- the average waiting time (number of ticks) of the customers who were served,
- the longest time any one customer waited, and
- the number of customers left in queues when the simulation terminated.

## ERRORS

The program may assume that the input values are as described above; it need not detect any errors.

## EXAMPLE

Below is an *abbreviated* script of a run of such a program. Each block of output lists the clock value, the times remaining in each teller's current transaction, and the arrival times of the customers waiting in the lines. For the first few ticks, each arriving customer is served immediately; these customers join queues but are dequeued without waiting.

Note that even with the same input values, your program will almost certainly produce different results, since a psuedo-random number generator's values can generally be used in several ways to get the effects we desire. **Examples like this one are intended to show what the program's output might look like, NOT to provide a standard to determine the correctness of your code. Your tests and examples must be different.**

```
Enter these parameters of the simulation:
The number of queue/server pairs: 4
The probability that a customer arrives in one tick (%): 80
The maximum duration of a transaction in ticks: 12
The duration of the simulation in ticks: 120
Enter a random number seed: 3
1     4
0
0
0

2     3
9
0
0

3     2
8
5
0   .
.
.
118   1  79 82 91 96 100 102 109 114 116
      6  80 87 88 97 101 103 110 115
      3     90 92 94 98 104 107 111 112
      12  93 95 99 105 108 113 118

119   0  79 82 91 96 100 102 109 114 116
      5  80 87 88 97 101 103 110 115
      2  90 92 94 98 104 107 111 112
      11  93 95 99 105 108 113 118 119

120   3  82 91 96 100 102 109 114 116
      4     80 87 88 97 101 103 110 115
      1  90 92 94 98 104 107 111 112
      10  93 95 99 105 108 113 118 119

68 customers waited an average of 16 ticks.
The longest time a customer waited was 41 ticks.
32 customers remain in the lines
```

## OTHER REQUIREMENTS

Implement the **Queue** abstract data type using either the sequential (array-based) implementation or the linked implementation.

## HINTS

Consider a function that returns the index of the "shortest" line; it will examine both the lengths of the queues and the remaining transaction times. Consider also a function that returns the total length of all the queues.

Use runs of your program to answer this **question**: If the probability that a customer arrives during one tick is 0.60, and the maximum transaction time is 15 ticks, what is the smallest number of lines and tellers the system can have and usually keep the average time customers wait below 10 ticks? Consider simulations at least 100 ticks long, and explain how you arrived at your conclusion.