Stack

Introduction -

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO(Last In First Out) or FILO(First In Last Out).

Some method associated to it
1) **push**: Adds an item in the stack. If the stack is full, then it is said to be an Overflow condition.
2) **pop**: Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition.
3) **top**: Returns top element of stack.
4) **size** - Returns size of stack
5) **isEmpty**: Returns true if stack is empty, else false

To know basics of stacks refer following links
https://www.geeksforgeeks.org/stack-data-structure-introduction-program/
https://www.hackerearth.com/practice/data-structures/stacks/basics-of-stacks/tutorial/

Uses -

1) Check balanced parentheses -
https://www.geeksforgeeks.org/check-for-balanced-parentheses-in-an-expression/`
2) Evaluation of prefix expression -
https://www.geeksforgeeks.org/evaluation-prefix-expressions/
3) Evaluation of postfix expression -
https://www.geeksforgeeks.org/stack-set-4-evaluation-postfix-expression/

http://www.spoj.com/problems/HISTOGRA/
The above question can be solved by [RMQ(Range minimum query)+Conquer And Divide] with complexity O(N logN) while it can be easily solved by stack with complexity O(N)

Solution using RMQ -
https://www.geeksforgeeks.org/largest-rectangular-area-in-a-histogram-set-1/
Solution using Stack -
https://www.geeksforgeeks.org/largest-rectangle-under-histogram

Quiz -

1) https://www.geeksforgeeks.org/data-structuare-gq/stack-gq/

Questions -

Easy :

1) https://www.codechef.com/problems/COMPILER and http://www.spoj.com/problems/ANARC09A/
2) http://www.spoj.com/problems/STPAR/
3) http://codeforces.com/contest/343/problem/B
4) http://www.spoj.com/problems/ONP/

Hard :

1) http://www.spoj.com/problems/MMASS/
2) http://codeforces.com/contest/5/problem/C
3) http://codeforces.com/contest/797/problem/C
4) https://www.hackerearth.com/practice/data-structures/stacks/basics-of-stacks/practice-problems/algorithm/monk-celebrating-checkpoint/

You can solve more problems on stack from -

1) HackerRank - https://www.hackerrank.com/domains/data-structures/stacks
2) HackerEarth - https://www.hackerearth.com/practice/data-structures/stacks/basics-of-stacks/practice-problems/

Queue

Introduction -

Like Stack, Queue is a linear structure which follows a particular order in which the operations are performed. The order is **F**irst **I**n **F**irst **O**ut (FIFO).The difference between stacks and queues is in removing. In a stack we remove the item the most recently added; in a queue, we remove the item the least recently added.

Some method associated to it
1) **Enqueue:** Adds an item to the queue. If the queue is full, then it is said to be an Overflow condition.
2) **Dequeue:** Removes an item from the queue. The items are popped in the same order in which they are pushed. If the queue is empty, then it is said to be an Underflow condition.
3) **Front:** Get the front item from queue.
4) **Rear:** Get the last item from queue.
5) **size** - Returns size of queue
6) **isEmpty**: Returns true if queue is empty, else false

Time Complexity - Time complexity for all above operation is O(1)

To know basics of queues refer following links
https://www.geeksforgeeks.org/queue-set-1introduction-and-array-implementation/
https://www.hackerearth.com/practice/data-structures/queues/basics-of-queues/tutorial/

Quiz -

https://www.geeksforgeeks.org/dataa-structure-gq/queue-gq/

Questions -

Refer following links for question based on simple queues
1) HackerEarth -
https://www.hackerearth.com/practice/data-structures/queues/basics-of-queues/practice-problems/
2) HackerRank -
https://www.hackerrank.com/domains/data-structures?filters%5Bsubdomains%5D%5B%5D=queues

Priority Queue

Introduction -

Priority Queue is an extension of queue with following properties.
1) Every item has a priority associated with it.
2) An element with high priority is dequeued before an element with low priority.
3) If two elements have the same priority, they are served according to their order in the queue.

Some method associated to it
1) **Enqueue:** Adds an item to the with queue pre determined priority. If the queue is full, then it is said to be an Overflow condition.
2) **Dequeue:** Removes an item from the queue having highest priority among all elements present in queue. The items are popped in the same order in which they are pushed. If the queue is empty, then it is said to be an Underflow condition.
3) **Front:** Get the element with highest priority.
4) **Rear:** Get the element with lowest priority.
5) **size** - Returns size of queue
6) **isEmpty**: Returns true if queue is empty, else false

Time Complexity -
Priority queue is implemented using heaps so we get front, rear, size and isEmpty operation are performed in O(1) while Enqueue and Dequeue operation are performed in O(log N) Read following article for further details about time compexity of operation in priority queue and heap
https://www.hackerearth.com/practice/notes/heaps-and-priority-queues/

Questions -

Easy :
1) http://www.spoj.com/problems/PQUEUE/
2)https://www.hackerearth.com/practice/data-structures/trees/heapspriority-queues/practice-problems/algorithm/roy-and-trending-topics-1/
3) http://codeforces.com/problemset/problem/982/B`
4) http://www.spoj.com/problems/PRO/

Hard :
1) www.spoj.com/problems/LAZYPROG/
2) https://www.codechef.com/JULY15/problems/MCHEF
3) http://codeforces.com/problemset/problem/20/C
4) http://codeforces.com/contest/377/problem/B

Refer following links for extra question based on priority queues
1) HackerEarth -
https://www.hackerearth.com/practice/data-structures/trees/heapspriority-queues/practice-problems/
2) HackerRank -
https://www.hackerrank.com/domains/data-structures?filters%5Bsubdomains%5D%5B%5D=heap
3) Contest -
https://www.hackerearth.com/challenge/competitive/code-monk-heaps-and-priority-queues-1/

Bit Manipulation

Introduction :-

Bit manipulation is the act of algorithmically manipulating bits or other pieces of data shorter than a word. Computer programming tasks that require bit manipulation include low-level device control, error detection and correction algorithms, data compression, encryption algorithms, and optimization.

Methods associated with it :-

1. And (&)
2. Or (|)
3. Xor (^)
4. Not (~)
5. Bit shifts
    a. Left shift (<<)
    b. Right shift (>>)

Tutorial :-

To know the basics of Bit Manipulation, refer following link.
https://www.hackerearth.com/practice/notes/bit-manipulation/

<u>**Questions**</u> :-

1. Easy :-

    a. https://www.hackerearth.com/practice/algorithms/dynamic-programming/bit-masking/practice-problems/algorithm/when-the-integers-got-upset/
    b. https://www.hackerearth.com/practice/basic-programming/bit-manipulation/basics-of-bit-manipulation/practice-problems/algorithm/monk-and-his-friend/
    c. https://www.hackerearth.com/practice/basic-programming/bit-manipulation/basics-of-bit-manipulation/practice-problems/algorithm/monks-choice-of-numbers-1/

2. Not that easy :-

    a. https://www.hackerrank.com/challenges/counter-game/problem
    b. https://www.hackerrank.com/challenges/the-great-xor/problem
    c. https://www.hackerrank.com/challenges/xor-se/problem
    HINT : You can find a pattern in this problem

You can solve more problems of Bit Manipulation from :-

1. Hackerrank
2. Hackerearth

Recursion

Introduction :-

Recursion in computer science is a method of solving a problem where the solution depends on solutions to smaller instances of the same problem (as opposed to iteration). The approach can be applied to many types of problems, and recursion is one of the central ideas of computer science.

Binary search, DFS (Graph theory), Merge sort, Quick sort are some examples of Recursion.

Tutorial :-

1. Hackerearth
2. Geeksforgeeks

Questions :-

1. Easy :-
    a. https://www.hackerearth.com/practice/basic-programming/recursion/recursion-and-backtracking/practice-problems/algorithm/a-tryst-with-chess/
    b. https://www.hackerearth.com/practice/basic-programming/recursion/recursion-and-backtracking/practice-problems/algorithm/n-queensrecursion-tutorial/
    c. https://www.hackerrank.com/challenges/the-power-sum/problem

2. Not that easy :-
    a. https://www.hackerearth.com/practice/basic-programming/recursion/recursion-and-backtracking/practice-problems/algorithm/jumpingjack-488ce744/
    b. https://www.hackerrank.com/challenges/repeat-k-sums/problem
    c. https://www.hackerrank.com/challenges/arithmetic-expressions/problem