Problem A:

Given is an ordered deck of n cards numbered 1 to n with card 1 at the top and card n at the bottom. The following operation is performed as long as there are at least two cards in the deck:

Throw away the top card and move the card that is now on the top of the deck to the bottom of the deck.

Your task is to find the sequence of discarded cards and the last, remaining card.
Each line of input (except the last) contains a number $2 \leq n \leq 52$. The last line contains 0 and this line should not be processed.

Sample:

Input:
7
19
2
0

Output:
Discarded cards: 1, 3, 5, 7, 4, 2,
Remaining card: 6

Discarded cards: 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 4, 8, 12, 16, 2, 10, 18, 14,
Remaining card: 6

Discarded cards: 1,
Remaining card: 2

Problem B:

WAP that will reverse a stack's elements within that stack. (You can't modify any code inside the stack structure. Use only conventional stack operations to solve the problem).

Hints: You can use temporary stacks.

Problem C:

WAP that will extract/remove the maximum element from a stack. (You can't modify any code inside the stack structure. Use only conventional stack operations to solve the problem).

Hints: You can use temporary stacks.

Problem D:

Implement a ***Stack data structure*** using two queues.

[Make it Menu driven: 1 means Push, 2 means Pop and so on….]

Problem E:

Write a program that will take a character string as input and then do following instructions:

>> search the string from 0 index.

>> During searching

>> If any character is English alphabet (a-z, A-Z) then push it into a stack.

>> If any character is Numeric digit then do a pop operation and show it on the screen.

>> For any other symbols or characters do nothing.

Sample input	***Sample output***
AB1D#$E69Z9A0KMN	BEDZA
nbmASgjash%U^&1971	Uhsa

Problem F:

Killing Vampires

Background:

You have given a game board in 2D matrix format where 'S' is the start point and 'E' is the end point. There is only a single path from 'S' to 'E'. In this path you will be face 3 types of Vampires which are marked as 'M', 'N' and 'O'. But to kill those vampires you need Blades. You will find 3 types of Blades on your way which are marked as 'A', 'B' and 'C'. Use Blade 'A' to kill Vampire 'M', Blade 'B' to kill Vampire 'N' and Blade 'C' to kill Vampire 'O'. You will have one chance to kill a vampire using a Blade. If you miss it, you will be killed by that vampire.

For carrying those Blades you have given a Bag of size 100 which is initially empty. For speed attack this bag works as Last in first out order for accessing blades. Now simulate the game starting from 'S' for different game board input.

Input:

First line contains the dimension (row and column) of the game board. In the game board 'X' means boundary wall, '-' means internal wall and '.' means free space. You can't steps on 'X' and '-' but you can steps on '.', 'A', 'B', 'C', 'M', 'N', 'O','E' and 'S'. You are not allowed to visit any place twice.

Output:

Follow sample output.

Sample inputs	***Sample outputs***
12 22 XXXXXXXXXXXXXXXXXXXXXX X--------------------X X---B..B...----------X X---M-----N..CC...A--X X---M-------------.--X X---O...C.N-------M--X X---------M..A----.--X X------------B----.--X X---------CAA.-----..-X	Blade found: C Blade found: A Blade found: A Blade found: B Blade found: A Enemy M is killed Enemy N is killed Blade found: C Enemy O is killed Enemy M is killed

X---------.--------E-X X---------S----------X XXXXXXXXXXXXXXXXXXXXX	Enemy M is killed Blade found: B Blade found: B Enemy N is killed Blade found: C Blade found: C Blade found: A Enemy M is killed Game complete.....
10 15 XXXXXXXXXXXXXXX X-------------X X-------------X X-------------X XS..A.--------X X----.--------X X----.--------X X----CB..NM.O-X X-----------E-X XXXXXXXXXXXXXXX	Blade found: A Blade found: C Blade found: B Enemy N is killed You are killed by enemy M

Problem G:

Killing Vampires-II

Background:

You have given a game board in 2D matrix format where 'S' is the start point and 'E' is the end point. There is only a single path from 'S' to 'E'. In this path you will be face 3 types of Vampires which are marked as 'M', 'N' and 'O'. But to kill those vampires you need Blades. You will find 3 types of Blades on your way which are marked as 'A', 'B' and 'C'. Use Blade 'A' to kill Vampire 'M', Blade 'B' to kill Vampire 'N' and Blade 'C' to kill Vampire 'O'. You will have one chance to kill a vampire using a Blade. If you miss it, you will be killed by that vampire.

Now your task is to start your journey from 'S' and

- Store all blades' information (blade types and matrix location) into a QUEUE
- Store all vampires' information (vampire types and matrix location) into another QUEUE.

Finally DEQUEUE all information and show it.

Input:

First line contains the dimension (row and column) of the game board. In the game board 'X' means boundary wall, '-' means internal wall and '.' means free space. You can't steps on 'X' and '-' but you can steps on '.', 'A', 'B', 'C', 'M', 'N', 'O','E' and 'S'. You are not allowed to visit any place twice.

Output:

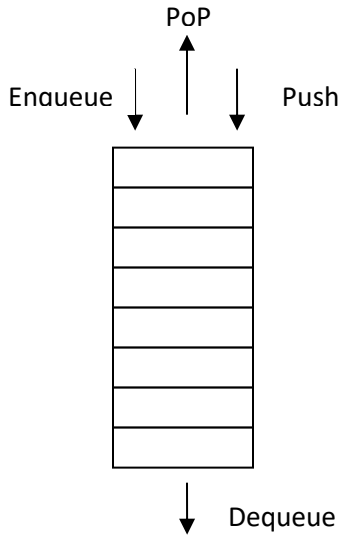
Follow sample output.

Sample inputs	***Sample outputs***
12 22 XXXXXXXXXXXXXXXXXXXXX X-------------------X X---B..B...----------X X---M-----N..CC...A--X X---M-------------.--X X---O...C.N-------M--X X---------M..A----.--X X------------B----.--X X---------CAA.----..-X X---------.--------E-X X---------S----------X XXXXXXXXXXXXXXXXXXXXX	Blade information : C 8 10 A 8 11 A 8 12 B 7 13 A 6 13 C 5 8 B 2 4 B 2 7 C 3 13 C 3 14 A 3 18 Vampire information : M 6 10 N 5 10 O 5 4 M 4 4 M 3 4 N 3 10 M 5 18
10 15 XXXXXXXXXXXXXXX X-------------X X-------------X X-------------X XS..A.--------X X----.--------X X----.--------X X----CB..NM.O-X X-----------E-X XXXXXXXXXXXXXXX	Blade information : A 4 4 C 7 5 B 7 6 Vampire information : N 7 9 M 7 10 O 7 12

Problem H:

A system has a 16 byte memory. Therefore it can hold only 8 integers. Initially the memory is empty. It can perform four operations. They are

1. Push
2. Pop
3. Enqueue and
4. Dequeue



Here **Push, Pop** operations maintain the rules of a simple stack and **Enqueue, Dequeue** operations maintains the rules of a simple queue. If the memory is full then the system discards any push or enqueue operation and if the memory is empty then it discards pop and dequeue operations. Now write a program using the concept of stack and queue to simulate the system.

Input: Test case will start with an input **n** that contains the numbers of operations. Next **n** line will have the operations. Specifications of operations are given below:

E x : Enqueue x // here x is an interger number
D : Dequeue
P x : Push x
O : Pop

Output: Output the final memory status of the system. If the memory is empty print a message "Memory is Empty".

Sample input	Sample output
12 D P 7 P 9 E 13 P 17 D E 19 D P 23 O O E 18	13 17 18