Page No.
Date

Name: Liza Lad Enrollment Number: 200303108087
Subject: DSA Code : 203105205

Assignment - 2
Ch-2 Stacks and Queue.

Q1 What is Stack ? List out different operation of it and write algorithm for any two operations.

Sol^n: A stack is an Abstract Data Type.. ~~Commonly used for most~~
It is named stack as it behaves like a real-world stack, for example:- a deck of cards or a pile of plates, etc.

⇒ Operations of stacks

Operations are limited.
1) push (add item to stack)
2) pop (remove top item from stack)
3) top (get top item without removing it)
4) clear
5) isEmpty
6) isFull
7) size?

⇒ Algorithm for Push Operation

Step 1: Checks if the stack is full.
Step 2: If the stack is full, produces an error and exit.
Step 3: If the stack is not full, increments top to point next empty space.

Page No.
Date

Step-4 :- Adds data element to the stack location, where top is pointing
Step-5 :- Returns success

⇒ Algorithm for pop operation

Step 1 - Checks if the stack is empty.
Step 2 - If the stack is empty, produces an error and exit.
Step 3 - If the stack is not empty, accesses the data element at which top is pointing.
Step 4 - Decreases the value of top by 1.
Step 5 - Returns success.

Q-2 Write a 'C' program or an algorithm to convert infix expression without parenthesis to postfix expression.

Soln: Algorithm

Let, 'x' is an arithmetic expression written in infix notation. This algorithm finds the equivalent postfix expression 'Y'.

(1) Push "(" onto Stack, and add ")" to the end of X.

(2) Scan X from left to right and repeat step 3 to 6 for each element of X until the stack is empty.

(3) If an operand is encountered, add it to Y.

(4) If a left parenthesis is encountered, push it onto stack.

(5) If an operator is encountered, then:

(1) Repeatedly pop from Stack and add to Y each operator (on the top of Stack) which has the same precedence as or higher precedence than operator.

(2) Add operator to stack.
[End of If]

(6) END.

Page No.
Date

Q-3 (1) Write an algorithm to reverse a string using stack.

Sol^n Step-1: String to char[]
Step-2: Create a stack
Step-3: Push all characters, one by one
Step-4: Then pop all characters, one by one and put into the char[].
Step-5: Finally, convert to the string.

(2) Write an algorithm to convert infix expression without parenthesis to post fix er

(2) Write an algorithm to check if an expression has balanced parenthesis using stack.

Sol^n Step-1: When any open symbol i.e, (, {, [is encountered, it will be pushed in the stack.

Step-2: If any close symbol i.e,), },] is encountered, any of the three can happen.

Step-3: The TOS (Top of stack) is checked, if the encountered close symbol matches with its open symbol, then open symbol which is at TOS is popped out.
(OR)
The TOS (Top of stack) is checked, if the encountered close symbol does not matches with its open symbol, then -1 is returned as there is no matching symbol.
(OR)

The TOS (Top of Stack) is checked. if the stack is empty, then -1 is returned as there is no open symbol in the stack.

Q4 What is recursion? What care should be taken in writing recursive function? Give example of any one recursive function. & What is Tower of Hanoi? Explain it with n = 3.

Sol^n Recursion is a programming Technique in which a function contains a function call to itself

A recursive algorithm must & change its state and move toward the base case to make sure that the function will terminate.

Example of any one recursive function

Direct recursion

Eg:- factorial(int x)
{
factorial (int x-1);
}

Tower Of Hanoi

It's a simple game concept which can be applied in stacks. The concept is that the bigger value should not be kept upon the smaller value.

Q.5 Translate the following string into polish notation and trace the content of stack.

A-(B/C+(D % E* F)/ G)* H

Soln

Infix Character Scanned	Stack	Postfix / Polish notation
A	C	A
-	(-	A
C	C-C	A
B	C-C	AB
/	C-C/	AB
C	C-C/	ABC
+	C-C+	ABC/
C	C-C+C	ABC/
D	C-C+C	ABC/D
%	C-C+C%	ABC/D
E	C-C+C%	ABC/DE
*	C-C+C%*	ABC/DE
F	C-C+C%*	ABC/DEF
)	C-C+	ABC/DEF*%
/	C-C+/	ABC/DEF*%
G	C-C+/	ABC/DEF*%G
)	C-	ABC/DEF*%G/+
*	C-*	ABC/DEF*%G/+
H	C-*	ABC/DEF*%G/+H
		ABC/DEF*%G/+H*-