Index

Name : LAKSHYA SAINI

Standard II^nd YEAR Division B

Roll No. 22EJCCS119 Subject DSA Assignment

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
1.	17-09-23	ASSIGNMENT-1	1	Deekh
2.	30-09-23	ASSIGNMENT-2	10	Deekh 05/10/23

Date 17/09/23
Page No. 1

ASSIGNMENT-1

Q1) What is stack? List the application of stacks.

Answer) A stack is a fundamental data structure in computer science and programming that follows the Last-In-First-Out (LIFO) principle. In a stack, the last element added is the first one to be removed. Think of it like a stack of plates where you can only add or remove plates from the top. Stacks are typically implemented using arrays or linked lists.

Applications of stack:-

-> A stack can be used for evaluating expressions consisting of operands and operators.

-> A stack can be used for backtracking, that is to check parenthesis matching in an expression.

-> It can also be used to convert one form of expression to another form.

-> It can be used for systematic memory management.

Q2) Define static and dynamic array.

Answer) Static Array:

In this types of array, memory is allocated at compile time having a fixed size of it, we can not alter or update the size of this array.

Dynamic Array:

In this type of array, memory is allocated at run time but not having a fixed size.

Q4) Explain the following terms:

(i) Infix Expression

Infix expressions are the most usual type of expression. This notation is typically employed when writing arithmetic expressions by hand. Moreover, in the infix expression, we place the operator between the two operands it operates on. The expression is of the form "A operator B" that is "A+B".

(ii) Prefix Expression

Prefix expressions, also known as Polish notation, place the operator before the operands. For example, in the expression "+AB", we place the "+" operator before the operands A and B, as demonstrated.

(iii) Postfix Expression

Postfix expressions are the ones in which we place the operator after the operands. The expressions are of the form "AB operator" that is "AB+".

Q5) Write down the algorithm for PUSH(), POP() and IsEmpty() operations.

Answer) Algorithm for PUSH()-

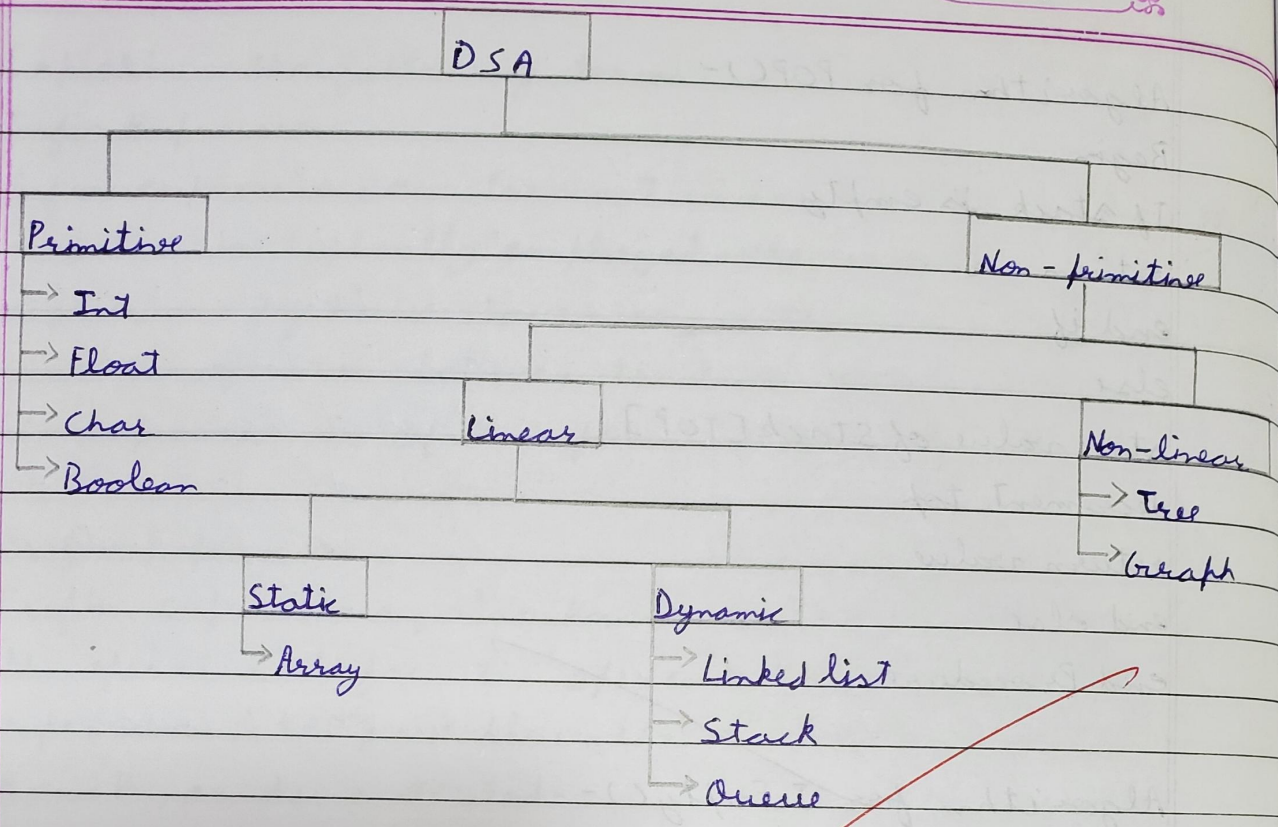
```
Begin
If stack is full
return
end if
else
increment of
Stack [TOP] assign
value
end else
end procedure
```

Algorithm for POP()-
Begin
If stack is empty
return
end if
else
store value of stack[TOP]
decrement top
return value
end else
end Procedure

Algorithm for IsEmpty()-
Begin
if top<1
return true
else
return false
end procedure

Q7) Define data structure. Explain the classification of DSA.

Answer) Data is a collection of facts and figures. Information is a collection of data. Data structure is a way to store and organize data so that it can be used efficiently. Here we have used the word efficiently which in terms of both space and time. They provide a means to store, retrieve, and manipulate data in a structured and organized manner.



- Primitive Data Structure

-> These are the most basic and fundamental data structures provided by programming languages. They include integer, float, character and boolean data types.

- Non-primitive Data Structure

-> Non-primitive data structure are those data structure which are derived from primitive data structures. Non-primitive data structure are further dived into linear and non-linear types.

Q6) Define Array and its types? With Example.

Answer) An array is a data structure in computer science that stores a collection of elements, each identified by an index or a key. The elements in an array are typically of the same data type, and they are stored in contiguous memory locations. Arrays provide efficient random access to elements based on their index, making them suitable for various applications where data needs to be organized in a linear or one-dimensional fashion.

☆ Types of arrays:

There are several types of arrays, depending on their characteristics and usage. Here are some common types:

① One-Dimensional Array:

A one dimensional array, also known as a vector or a linear array, is the simplest form of an array. It represents a list of elements in a single line or row.

For example, numbers[5] = {1, 2, 3, 4, 5}

② Two-Dimensional Array:

A two dimensional array, also known as a matrix, consists of rows and columns, forming a grid-like structure.

For example, int twodimen[4][3];

③ Multi-Dimensional Array:

Multi-dimensional arrays extend the concept of two-dimensional arrays to more than two dimensions. They can have three or more indices.

For example, float x[2][4][3];

Q8) Differentiate between linear and non-linear data structure.

Answer)

S.No.	Parameter	Linear Data Structure	Non-linear Data Structure
1.	Arrangement of data element	In a linear data structure the data elements connect to each other, Sequentially. a user can transverse each element through a single run.	In a non-linear data structure the data elements connect to each other hierarchically. Thus, they are present at various levels.
2.	Complexity of implementation	The linear data structures are comparatively easier to implement.	The non-linear data structures are comparatively difficult to implement and understand as compared to the linear data structure.
3.	Levels	A user can find all of the data elements at a single level in a linear data structure.	One can find all the data elements at multiple levels in a non-linear data structure.
4.	Traversal	You can traverse a linear data structure in a single run.	It is not easy to traverse non-linear data structures. The users need multiple runs to traverse them completely.

Q4) Translate infix expression into its equivalent postfix expression (with and without stack)-

(i) (A-B)*(D/E)

(A-B)*DE/

A-B BDE/*

AB-DE/*

(iii) A*(B+c*D)+E

A*(B+cD)*+E

A(B+cD)**E+

ABcD *+*E+

(ii) A+B/c-D^E-F

A+B/c-DE^-F

A+Bc/-DE^F-

ABc/+DE^-F-

(iv) A/B-(c+D)*E/F^G

AB/-(c+D)*E/FG^

AB/-(c+D)E*FG^/

AB/(c+D)E*FG^/-

AB/cD+E*FG^/-

Q11) Transform the following into infix-

(i) +*ABc

+(A*B)c

((A*B)+c)

(ii) ABc*+

A(B*c)+

(A+(B*c))

(iii) ABC*/
A (B*C)/
(A/(B*C))

(iv) ABC*+DE/F*-
A(B*C)+D/EF*-
(A+(B*C)) D/E*F-
(A+(B*C))-((D/E)*F)

Q12) Evaluate the following postfix expressions:-

(i) 987*+
9(8*7)+
(8*7)+9
56+9
=65

(ii) 76+44 10-15+
76+44 10*-15+
764410*+-15+
764410*+51-+

Q3) Write down the recursive algorithm to solve tower of Hanoi problem.

Ans) The Tower of Hanoi is a classic recursive problem that involves moving a stack of disks from one rod to another with the constraint that no disk can be placed on top of a smaller disk.

Step 1) If n=1, move the single disk from A to C and return.

Step 2) If n>1, move the top n-1 disks from A to B using C as temporary.

Step 3) Move the remaining disk from A to C.

Step 4) Move the n-1 disk from B to C, using A as temporary.

The algorithm continues to break down the problem into smaller subproblems until it reaches the base case, where it can perform the actual disk move. This recursive approach ensures that the Tower of Hanoi problem is solved correctly, following the constraints of not placing larger disks on top of smaller ones.