IARE
INSTITUTE OF AERONAUTICAL ENGINEERING
(An Autonomous Institute affiliated to JNTUH, Hyderabad)
Dundigal, Hyderabad - 500 043

LABORATORY WORK BOOK

Name of the Student : Racherla Santhosh

Class IT-B Semester 03

Course Code : ACSD11 Course Name : DS laboratory

Name of the Course Faculty Ms. K. Laxminarayanamma Faculty ID : IARE 10033

Exercise Number : 05 Week Number : 05 Date : 01/10/2024

Roll Number									
2	3	9	5	1	A	1	2	G	3

S. No.	Exercise Number	EXERCISE NAME	MARKS AWARDED					
			Aim/ Preparation	Algorithm / Procedure Performance in the Lab	Source Code Calculations and Graphs	Program Execution Results and Error Analysis	Viva - Voce	Total
			4	4	4	4	4	20
1	5.1	Implementation Of Stack						
2	5.2	Balanced parenthesis checking						
3	5.3	Evaluation Of PF Expression						
4	5.4	Infix To post fix						
5	5.5	Reverse A Stack						
6			4	4	4	4	4	20
7								
8								
9								
10								
11								
12								

Santhosh
Signature of the Student

[illegible]
Signature of the Faculty

5. Stack.

5.1 Implementation Of Stack :-

AIM :- Write a Program On Implementation Of Stack.

PROGRAM :-

```
import java.util.Scanner;
import java.util.Stack;
class Stack {
  Public Static void main (String args[]) {
    Stack <Integer> Stack = new Stack <>();
    Scanner bk = new Scanner (System.in);
    While (true) {
      System.out.println ("1. push");
      System.out.println ("2.pop");
      System.out.println ("3. Peek");
      System.out.println ("4. Check if Empty");
```

ROLL NUMBER :

```
System.out.println ("5. size");
System.out.println ("6. Exit");
System.out.println ("Enter your choice");
int c = bK.nextInt();
Switch (c) {
  case 1 :
    System.out.print (" Enter element to push:");
    int e = bK.nextInt();
    Stack.push(e);
    System.out.println(e+" Pushed to Stack");
    break;
  case 2 :
    if (! Stack.isEmpty()) {
      int PE = Stack.pop();
      System.out.println (PE + " Popped from Stack");
  }
  else {
    System.out.println (" Stack is Empty");
  }
  break;
```

ROLL NUMBER :

```
case 3 :
  if (! stack. isEmpty ()) {
    System. out. println (" Top element.: " + stack . peek());
  } else {
    System. out. println ("Stack is Empty");
  }
  break ;
case 4 :
  System. out. println (" Is Stack Empty ? "+ stack . is
                                  Empty()) ;
  break;
case 5 :
  System. out. println (" stack size :"+ stack . size())
  break;
case 6 :
  System . out. println ("Exiting ");
  bk. close() ;
  return;
default :
  System. out. println ("Invalid choice ... try again");
}
}
```

ROLL NUMBER :

RESULT :-

INPUT : 1. push
2. pop
3. Peek
4. Check if Empty
5. Size
6. Exit
Enter your choice : 1

OUTPUT : Enter Element To Push : 7
7 pushed to Stack.

5.2 Balanced Parenthesis Checking :-

AIM :- Given an Expression String, Write a JAVA Program to find whether a given String has balanced Parenthesis or not.

PROGRAM :-

```
import java.util.* ;
class Bp {
Public Static void main (String[] args) {
    Scanner bk = new Scanner (System.in);
```

ROLL NUMBER :

```
String g = sk. nextLine();
boolean isBalanced = (BP (g));
if (is Balanced) {
  System. out. println ("True");
} else {
  System. out. println (" False");
}
}
Public static boolean GBP(String g) {
 Stack <> Stack = new Stack <> ();
 for (char ch: g. to Char Array()) {
  if (ch == '(') {
   Stack. push (ch);
  } else if (ch == ')') {
    if (Stack . is Empty()) {
         return False;
    }
   Stack. pop();
  }
}
return Stack. is Empty (); }}
```

RESULT :-

INPUT :- "{(a+b)* [c-d]}"

OUTPUT :- False

5.3 Evaluation Of Postfix Expression :-

AIM :- Given a Postfix Expression, Write a Program that the given task is evaluate the Postfix Expression.

PROGRAM :-

```
import java.util.* ;
class postfix {
 static int EPf (String exp) {
  Stack < Integer > Stack = new Stack <> () ;
   for ( int i=0; i< exp. length(); i++) {
    char c = exp. charAt (i);
    if (Character. isDigit (c)) {
      Stack. push ( c-'0'); }
    else {
```

ROLL NUMBER :

```
      int  val 1  =  Stack . pop() ;
      int  val 2  =  Stack . pop() ;
      Switch (c) {
        case '+' :
          Stack . push (val 2 + val 1) ;
          break ;
        case '-' :
          Stack . push (val 2 - val 1) ;
          break ;
        case '/' :
          Stack . push (val2 / val 1) ;
          break ;
        case '*' :
          Stack . push (val 2 * val 1) ;
          break ;
      }
    }
  }
  return Stack . POP () ;
  }
  Public Static void main ( String args[] ) {
    Scanner bK = new Scanner (System . in) ;
```

ROLL NUMBER :

```
        System.out.print ("Enter a postfix Expression");
        String exp = bk.nextLine();
        System.out.println ("postfix evaluation : "+ Epf (exp));
    }
}
```

RESULT :-

INPUT : Str = " 100 200 + 2 / 5 * 7 +"

OUTPUT : 757

5.4 Infix To Postfix Expression Conversion :-

AIM :- Write a Program to Convert a given Infix Expression into Postfix Expression.

PROGRAM :-

```
import java.util.* ;
class Itp {
  static int IP (char ch) {
    Switch (ch) {
      case '+' :
      case '-' :
      return 1 ;
```

```
        case '*' :
        case '/' :
            return 2 ;
        case '^' ;
            return 3 ;
    }
    return -1 ;
}
Static String ItP ( String exp) {
    String Builder result = new String Builder () ;
    Stack < character > Stack = new Stack <>() ;
    for (int i=0 ; i< exp.length(); i++) {
        char c = exp. CharAt (i) ;
    if (Character . isLetter or Digit (c))
        result. append (c) ;
    else if   (c = = '(')
        Stack . push (c) ;
    else if (c = ')') {
        while (! Stack . isEmpty() && Stack . peek()!
                = '(')
        result . append (Stack . pop ()) ;
```

ROLL NUMBER :

```
    Stack . pop ( ) ;
  } else {
    While  (! Stack . isEmpty ()  &&  Ip (c) <= Ip
            ( Stack . peek()) {
      Result. append  (Stack . pop()) ;
  }
  Stack . push (c) ;
 }}
  While  (! Stack . isEmpty()) {
     Result. append (Stack . Pop ()) ;
  }
  return  result . to String () ;
 }
 Public   Static  void   main ( String  args[]) {
   Scanner  sc  =  new  Scanner ( System. in);
   String  exp  =  sc. next Line( ) ;
   System . out . println ("Ip (exp)) ;
  }
 }
```

RESULT :- INPUT : A + B* C + D   OUTPUT : ABC* + D+

ROLL NUMBER :

5.5 ## Reverse A Stack :-

AIM :- Write a Program On Reversing a Stack.

PROGRAM :-

```
import java.util.*;
class Reverse {
  static void IB (stack < Integer > stack, int item) {
    if (stack.isEmpty()) {
      stack.push (item);
      return;
    }
   int top = stack.pop();
   IB (stack, item);
   stack.push (top);
 }
 static void RS (stack < Integer > stack) {
   if (stack.isEmpty())
     return;
   int top = stack.pop();
   RS (stack).
   IB (stack.top);
}
```

ROLL NUMBER :

```
Public Static void main ( String[] args) {
  Scanner  sc = new Scanner ( System . in);
  Stack <Integer> stack = new Stack <x>);
  System. out. print (" Enter Size Of Stack : ");
  int n = sc. nextInt() ;
  System. out. println (" Enter Elements: ");
  for (int i=0; i<n; i++) {
     Stack. push ( sc. nextInt ());
  }
  Rs ( Stack) ;
  System. out. println ("Reversal Stack : " +
                               Stack) ;
  }
}
```

RESULT :-

INPUT : Elements = [1, 2, 3, 4, 5]

OUTPUT :
```
Original Stack        Stack After Reversing
   5                      1
   4                      2
   3                      3
   2                      4
   1                      5
```

ROLL NUMBER :

VIVA VOCE :-

1) What is a Stack ?

A) A Stack is a linear data Structure that Stores items in a Last-In / First-Out (LIFO) or First-In / Last-Out (FILO) manner. In Stack, a new element is added at one end and an element is removed from that end only. The Insert & Delete operations are often called push and POP.

2) What are Infix and Postfix Expressions ?

A) Infix Expression :- Where the Operator is placed between Operands like "A + B".

Postfix Expression :- Where the Operator is placed after the operands like "AB+".

3) What do "Bottom Insertion ()" Method work as ?

A) This Method Append Element at the bottom

ROLL NUMBER :

of the Stack and Bottom Insertion accept two values as an argument first is Stack and the Second is elements, this is a recursive method.

4) How do "Reverse()" Method will work ?

A) The method is reverse elements of the Stack, this method accept Stack as an argument Reverse() is also a Recursive() function. Reverse() is invoked BottomInsertion() method for completing the reverse operation on the Stack.

[illegible]