Name: Vishwajit Gadale
Roll No: 15
FYMCA

PRACTICAL NO:- 01

Aim(A): Implementation of different searching.

i)Linear Search

Source Code:

```
import java.util.Scanner;
public class LinearSearch {
   public static void main(String[] args) {
      int i, x, n;
      int flag = 0;
      Scanner scanner = new Scanner(System.in);
      System.out.println("rollNo-15");
      System.out.println("How many numbers you want to enter in the array?");
      n = scanner.nextInt();
      // Declare array
      int[] a = new int[n];
      // Input array elements
      System.out.println("Enter Elements:");
      for (i = 0; i < n; i++) {
         a[i] = scanner.nextInt();
      }
      // Input the number to search for
      System.out.println("Enter number which you want to search:");
      x = scanner.nextInt();
      // Linear search
```

```
        for (i = 0; i < n; i++) {
            if (a[i] == x) {
                flag = 1;
                break;
            } }
        // Output result
        if (flag == 1) {
            System.out.println("Element Found!");
        } else {
            System.out.println("Element not found!");
        }
        scanner.close();
    }}
```

Output :

```
rollNo-15
How many numbers you want to enter in the array?
5
Enter Elements:
12 23 34 45 56
Enter number which you want to search:
34
Element Found!
```

```
rollNo-15
How many numbers you want to enter in the array?
5
Enter Elements:
12 23 34 45 56
Enter number which you want to search:
44
Element not found!
```

ii)Binary Search

Source Code:

```
import java.util.Scanner;
public class BinarySearch {
 // Binary search method
 public static int binarySearch(int[] arr, int l, int r, int x) {
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
if (r >= l) {
int mid = l + (r - l) / 2;
// Check if x is present at mid
if (arr[mid] == x) {
return mid;
}
// If x is smaller than mid, search in the left half
if (arr[mid] > x) {
return binarySearch(arr, l, mid - 1, x);
}
// If x is larger than mid, search in the right half
return binarySearch(arr, mid + 1, r, x);
}
// Return -1 if the element is not present in the array
return -1;
}
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
System.out.println("Roll no.15");
// Initialize the array
int[] arr = new int[5];
// Taking user input for array elements
for (int i = 0; i < 5; i++) {
System.out.print("Enter element " + (i + 1) + ": ");
arr[i] = scanner.nextInt();
}
// Displaying the elements
System.out.println("Elements are:");
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
for (int i = 0; i < 5; i++) {
System.out.println(arr[i]);
}
System.out.print("Enter the element to search: ");
int x = scanner.nextInt();
// Calling binary search and displaying the result
int n = arr.length;
int result = binarySearch(arr, 0, n - 1, x);
if (result == -1) {
System.out.println("Element is not present in the array");
} else {
System.out.println("Element is present at index " + result);
}
scanner.close();
}}
```

Output:

```
Roll no.15
Enter element 1: 32
Enter element 2: 21
Enter element 3: 54
Enter element 4: 65
Enter element 5: 76
Elements are:
32
21
54
65
76
Enter the element to search: 54
Element is present at index 2
```

```
Roll no.15
Enter element 1: 32
Enter element 2: 21
Enter element 3: 54
Enter element 4: 65
Enter element 5: 76
Elements are:
32
21
54
65
76
Enter the element to search: 44
Element is not present in the array
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

Aim(B): Implementation of different sorting techniques.

i)Bubble Sort

Source Code:

```
import java.util.Scanner;
public class BubbleSort {
 public static void main(String[] args) {
 System.out.println("roll no. 15.");
 Scanner scanner = new Scanner(System.in);
 System.out.println("How many numbers you want to enter?");
 int n = scanner.nextInt();
 int[] a = new int[n];
 System.out.println("Enter Elements:");
 for (int i = 0; i < n; i++) {
 a[i] = scanner.nextInt();
 }
 // Bubble sort algorithm
 for (int i = 0; i < n - 1; i++) {
 for (int j = 0; j < n - i - 1; j++) {
 if (a[j] > a[j + 1]) {
 // Swap elements
 int temp = a[j];
 a[j] = a[j + 1];
 a[j + 1] = temp;
 }}}
 // Print sorted array with spaces
 System.out.println("Sorted array is:");
```

```
for (int i = 0; i < n; i++) {
System.out.print(a[i]);
if (i != n - 1) {
System.out.print(" "); // Print space after every number except the last one
} }
System.out.println(); // Add a newline after the sorted array
scanner.close();
}}
```

Output :

```
roll no. 15.
How many numbers you want to enter?
5
Enter Elements:
12 98 87 76 65
Sorted array is:
12 65 76 87 98
```

ii) Insertion Sort

Source Code:

```
import java.util.Scanner;
public class InsertionSort {
 // Method to display the array
 public static void display(int[] array, int size) {
 for (int i = 0; i < size; i++) {
 System.out.print(array[i] + " ");
 }
 System.out.println();
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
}
// Insertion sort algorithm
public static void insertionSort(int[] array, int size) {
int key, j;
for (int i = 1; i < size; i++) {
key = array[i];
j = i;
// Move elements of array[0..i-1] that are greater than key to one position ahead
while (j > 0 && array[j - 1] > key) {
array[j] = array[j - 1];
j--;
}
array[j] = key;
} }
public static void main(String[] args) {
// Print roll number at the start of the output
System.out.println("roll no.15");
Scanner scanner = new Scanner(System.in);
// Read the number of elements
System.out.print("Enter the number of elements: ");
int n = scanner.nextInt();
// Create the array to hold the elements
int[] arr = new int[n];
// Read the elements into the array
System.out.println("Enter elements:");
for (int i = 0; i < n; i++) {
arr[i] = scanner.nextInt();
}
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
// Display the array before sorting
System.out.print("Array before Sorting: ");
display(arr, n);
// Perform the insertion sort
insertionSort(arr, n);
// Display the array after sorting
System.out.print("Array after Sorting: ");
display(arr, n);
scanner.close();
}}
```

Output:

```
roll no.15
Enter the number of elements: 5
Enter elements:
43 56 32 12 98
Array before Sorting: 43 56 32 12 98
Array after Sorting: 12 32 43 56 98
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

iii) Selection Sort

Source Code:

```
import java.util.Scanner;
public class SelectionSort {
 // Method to display the array
 public static void display(int[] array, int size) {
 for (int i = 0; i < size; i++) {
 System.out.print(array[i] + " ");
 }
 System.out.println();
 }
 // Method to perform selection sort
 public static void selectionSort(int[] array, int size) {
 int imin, temp;
 for (int i = 0; i < size - 1; i++) {
 imin = i;
 for (int j = i + 1; j < size; j++) {
 if (array[j] < array[imin]) {
 imin = j;
 } }
 // Swap the elements
 temp = array[i];
 array[i] = array[imin];
 array[imin] = temp;
 } }
 public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
// This line ensures your roll number is printed at the start
System.out.println("roll no.15");
System.out.println("Enter the number of elements: ");
int n = sc.nextInt();
int[] arr = new int[n];
System.out.println("Enter elements:");
// Accepting array elements from the user
for (int i = 0; i < n; i++) {
arr[i] = sc.nextInt();
}
// Display the array before sorting
System.out.print("Array before Sorting: ");
display(arr, n);
// Performing selection sort
selectionSort(arr, n);
// Display the array after sorting
System.out.print("Array after Sorting: ");
display(arr, n);
} }
```

Output :

```
roll no.15
Enter the number of elements:
5
Enter elements:
32 54 65 12 87
Array before Sorting: 32 54 65 12 87
Array after Sorting: 12 32 54 65 87
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

iv) Shell Sort

Source Code:

```
import java.util.Scanner;
public class ShellSort {
 public static void main(String[] args) {
 System.out.println("roll no.15");
 Scanner scanner = new Scanner(System.in);
 // Read the size of the array
 System.out.print("Enter the size of the array: ");
 int n = scanner.nextInt();
 // Create the array and read its elements
 int[] a = new int[n];
 System.out.print("Enter the elements of the array: ");
 for (int i = 0; i < n; i++) {
 a[i] = scanner.nextInt();
 }
 // Print array before sorting
 System.out.print("Array elements before sorting: ");
 for (int i = 0; i < n; i++) {
 System.out.print(a[i] + " ");
 }
 System.out.println();
 // Shell Sort Logic
 for (int gap = n / 2; gap > 0; gap /= 2) {
 for (int i = gap; i < n; i++) {
 int temp = a[i];
 int j = i;
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
// Move elements of the array that are greater than temp to the gap distance ahead
while (j >= gap && a[j - gap] > temp) {
a[j] = a[j - gap];
j -= gap;
}
a[j] = temp;
}}
// Print array after sorting
System.out.print("Array after sorting: ");
for (int i = 0; i < n; i++) {
System.out.print(a[i] + " ");
}
// Close the scanner
scanner.close();
} }
```

Output :

```
roll no.15
Enter the size of the array: 5
Enter the elements of the array: 12 23 43 23 31
Array elements before sorting: 12 23 43 23 31
Array after sorting: 12 23 23 31 43
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

PRACTICAL NO:- 02

Aim :- Perform various hashing techniques with Linear Probe as collision resolution

Source Code:

```
import java.util.Scanner;
public class HashingLinearProbing {
 // Function to perform linear probing search in the hash table
 public static int hashSearch(int[] hashTable, int x, int n) {
 int index = x % n;
 int start = index; // To prevent infinite loop if the entire table is searched
 if (hashTable[index] == x) {
 return index;
 } else if (hashTable[index] == -1) {
 return -1;
 } else {
 do {
 index = (index + 1) % n; // Linear probing
 if (hashTable[index] == x) {
 return index;
 } else if (hashTable[index] == -1) {
 break;
 }
 } while (index != start);
 return -1; // Element not found
 }  }
 public static void main(String[] args) {
 Scanner scanner = new Scanner(System.in);
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
int[] hashTable = new int[10]; // Hash table with size 10
int i, x, index, k;
// Initialize the hash table with -1 (indicating empty slots)
for (i = 0; i < 10; i++) {
hashTable[i] = -1;
}
// Input 7 elements and insert them into the hash table using linear probing
System.out.println("Roll no: 15");
System.out.println("Hashing using linear probing\n");
System.out.println("Hash table creation\n");
for (i = 1; i <= 7; i++) {
System.out.print("\nEnter a number: ");
x = scanner.nextInt();
index = x % 10; // Hash function (modulo)
// Linear probing to find an empty slot
while (hashTable[index] != -1) {
index = (index + 1) % 10;
}
hashTable[index] = x; // Insert the element at the found index
}
// Display the hash table
System.out.println("\nHASH TABLE");
for (i = 0; i < 10; i++) {
System.out.print(i + " ");
}
System.out.println();
for (i = 0; i < 10; i++) {
System.out.print(hashTable[i] + " ");
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
}
// Search loop
do {
System.out.print("\nElement to be searched (enter -1 to stop): ");
x = scanner.nextInt();
if (x >= 0) {
k = hashSearch(hashTable, x, 10);
if (k >= 0) {
System.out.println(x + " is present at hash[" + k + "]");
} else {
System.out.println(x + " is not present");
} }
} while (x >= 0);
scanner.close();
} }
```

Output :

```
Roll no: 15
Hashing using linear probing

Hash table creation


Enter a number: 12
Enter a number: 32
Enter a number: 54
Enter a number: 32
Enter a number: 21
Enter a number: 89
Enter a number: 43
```

```
HASH TABLE
0 1 2 3 4 5 6 7 8 9
-1 21 12 32 54 32 43 -1 -1 89
Element to be searched (enter -1 to stop): 89
89 is present at hash[9]

Element to be searched (enter -1 to stop): 54
54 is present at hash[4]

Element to be searched (enter -1 to stop): -1
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

PRACTICAL NO:- 03

Aim:- Implementation of Using Array

i)Stacks

Source Code:

```
import java.util.Scanner; class Stack {
   private static final int MAX = 10; private int[] a;
   private int top; public Stack() {
   a = new int[MAX]; top = -1;
   }
   public boolean isEmpty() { return top == -1;
   }
   public boolean isFull() { return top == MAX - 1;
   }
   public void push(int value) { if (isFull()) {
   System.out.println("Stack overflow, Cannot push " + value + "."); return;
   }
   a[++top] = value;
   System.out.println(value + " pushed onto stack");
   }
   public void pop() { if (isEmpty()) {
   System.out.println("Stack is empty. Cannot pop."); return;
   }
   System.out.println(a[top--] + " popped from stack");
   }
   public void peek() { if (isEmpty()) {
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
System.out.println("Stack is empty."); return;
}
System.out.print("The elements are: "); for (int i = 0; i <= top; i++) {
System.out.print(a[i] + " ");
}
System.out.println();
}
}
public class Main {
public static void main(String[] args) { System.out.println("Roll No.15 "); Stack stack = new Stack();
stack.push(10); stack.push(20); stack.peek();
stack.pop();
stack.peek();
stack.pop();
stack.pop();
}
}
```

Output :

```
Roll No.15
10 pushed onto stack
20 pushed onto stack
The elements are: 10 20
20 popped from stack
The elements are: 10
10 popped from stack
Stack is empty. Cannot pop.
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

ii) Ordinary Queue

Source Code:

```
import java.util.Scanner;
class Queue {
   private int front, rear; private int[] q;
   private static final int max = 10;
   public Queue() { front = rear = -1; q = new int[max];
   }
   public void enqueue(int x) { if (rear == max - 1) {
   System.out.println("Queue is Full!");
   } else {
   if (front == -1 && rear == -1) { front = rear = 0;
   } else {
   rear++;
   }
   q[rear] = x;
   System.out.println(x + " enqueued to queue");
   }  }
   public int dequeue() { int x = -1;
   if (front == -1) { System.out.println("Queue is Empty!");
   } else {
   if (front == rear) { x = q[front]; front = rear = -1;
   } else {
   x = q[front]; front++;
   }  }
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
    return x;
    }
    public void peek() { if (front == -1) {
    System.out.println("Queue is Empty!");
    } else {
    System.out.print("Queue Elements are: "); for (int i = front; i <= rear; i++) {
    System.out.print(q[i] + " ");
    }
    System.out.println();
    }
    } }
    public class Queues {
    public static void main(String[] args) { System.out.print(" Roll No.15 \n"); Queue q = new
Queue();
    Scanner scanner = new Scanner(System.in); int ch, x;
    do {
    System.out.print("1. Enqueue    2. Dequeue  3. Display Queue    4. Exit: "); ch =
scanner.nextInt();
    switch (ch) { case 1:
    System.out.print("Enter the value to be inserted: "); x = scanner.nextInt();
    q.enqueue(x); break;
    case 2:
    x = q.dequeue(); if (x != -1) {
    System.out.println(x + " dequeued!");
    }
    break; case 3:
    q.peek(); break;
    case 4:
    return; default:
```

```
System.out.println("Invalid choice. Exiting."); return;
}
} while (ch != 4); scanner.close();
} }
```

Output :

```
 Roll No.15
1. Enqueue      2. Dequeue      3. Display Queue        4. Exit: 1
Enter the value to be inserted: 21
21 enqueued to queue
1. Enqueue      2. Dequeue      3. Display Queue        4. Exit: 1
Enter the value to be inserted: 12
12 enqueued to queue
1. Enqueue      2. Dequeue      3. Display Queue        4. Exit: 1
Enter the value to be inserted: 54
54 enqueued to queue
1. Enqueue      2. Dequeue      3. Display Queue        4. Exit: 3
Queue Elements are: 21 12 54
1. Enqueue      2. Dequeue      3. Display Queue        4. Exit: 2
21 dequeued!
1. Enqueue      2. Dequeue      3. Display Queue        4. Exit: 3
Queue Elements are: 12 54
1. Enqueue      2. Dequeue      3. Display Queue        4. Exit: 4
PS D:\Documents\MCA\ADS Lab\Practicals\ADS Practicals>
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

iii) Circular Queue

Source Code:

```
import java.util.Scanner;
public class CircularQueue {
 static int[] cqueue = new int[5];
 static int front = -1, rear = -1, n = 5;
 // Method to insert an element in the queue
 public static void insertCQ(int val) {
 if ((front == 0 && rear == n - 1) || (front == rear + 1)) {
 System.out.println("Queue Overflow");
 return;
 }
 if (front == -1) {
 front = 0;
 rear = 0;
 } else {
 if (rear == n - 1)
 rear = 0;
 else
 rear = rear + 1;
 }
 cqueue[rear] = val;
 }
 // Method to delete an element from the queue
 public static void deleteCQ() {
 if (front == -1) {
 System.out.println("Queue Underflow");
 return;
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
}
System.out.println("Element deleted from queue is: " + cqueue[front]);
if (front == rear) {
front = -1;
rear = -1;
} else {
if (front == n - 1)
front = 0;
else
front = front + 1;
} }
// Method to display elements of the queue
public static void displayCQ() {
int f = front, r = rear;
if (front == -1) {
System.out.println("Queue is empty");
return;
}
System.out.println("Queue elements are:");
if (f <= r) {
while (f <= r) {
System.out.print(cqueue[f] + " ");
f++;
}
} else {
while (f <= n - 1) {
System.out.print(cqueue[f] + " ");
f++;
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
 }
 f = 0;
 while (f <= r) {
 System.out.print(cqueue[f] + " ");
 f++;
 } }
 System.out.println();
 }
 public static void main(String[] args) {
 System.out.println("Roll no: 15");
Scanner scanner = new Scanner(System.in);
 int ch, val;
 do {
 System.out.println("1) Insert");
 System.out.println("2) Delete");
 System.out.println("3) Display");
 System.out.println("4) Exit");
 System.out.print("Enter choice: ");
 ch = scanner.nextInt();
 switch (ch) {
 case 1:
 System.out.print("Input for insertion: ");
 val = scanner.nextInt();
 insertCQ(val);
 break;
 case 2:
 deleteCQ();
 break;
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
case 3:
displayCQ();
break;
case 4:
System.out.println("Exit");
break;
default:
System.out.println("Incorrect choice!"); }
} while (ch != 4);
scanner.close();
} }
```

Output :

```
Roll no: 15
1) Insert
2) Delete
3) Display
4) Exit
Enter choice: 1
Input for insertion: 15
1) Insert
2) Delete
3) Display
4) Exit
Enter choice: 1
Input for insertion: 87
1) Insert
2) Delete
3) Display
4) Exit
Enter choice: 1
Input for insertion: 41
1) Insert
2) Delete
3) Display
4) Exit
Enter choice: 3
Queue elements are:
15 87 41
1) Insert
2) Delete
3) Display
4) Exit
Enter choice: 2
Element deleted from queue is: 15
1) Insert
2) Delete
3) Display
4) Exit
Enter choice: 3
Queue elements are:
87 41
1) Insert
2) Delete
3) Display
4) Exit
Enter choice: 4
Exit
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

PRACTICAL NO:- 04

Aim :- Implementation of Stack Applications like i)Infix to Postfix

Source Code:

```
import java.util.Scanner;
class InfixToPostfixWithoutImport { static final int MAX = 100;
public static int precedence(char c) { if (c == '^') return 3;
if (c == '*' || c == '/') return 2; if (c == '+' || c == '-') return 1; return -1;
}
public static boolean isOperand(char c) { return Character.isLetterOrDigit(c);
}
public static String infixToPostfix(String infix) { char[] stack = new char[MAX];
int top = -1;
StringBuilder postfix = new StringBuilder(); for (int i = 0; i < infix.length(); i++) {
char currentChar = infix.charAt(i); if (isOperand(currentChar)) {
postfix.append(currentChar);
}
else if (currentChar == '(') { stack[++top] = currentChar;
}
else if (currentChar == ')') {
while (top != -1 && stack[top] != '(') { postfix.append(stack[top--]);
}
top--;
}
else {
while (top != -1 && precedence(stack[top]) >= precedence(currentChar)) {
postfix.append(stack[top--]);
```

```
}
stack[++top] = currentChar;
} }
while (top != -1) { postfix.append(stack[top--]);
}
return postfix.toString();
}
public static void main(String[] args) { System.out.print("Roll No.15    \n");
Scanner scanner = new Scanner(System.in); System.out.print("Enter the infix expression: ");
String infix = scanner.nextLine();
String postfix = infixToPostfix(infix);
System.out.println("Postfix Expression: " + postfix);
} }
```

Output :

```
Roll No.15
Enter the infix expression: (A+B-C)*D
Postfix Expression: AB+C-D*
PS D:\Documents\MCA\ADS Lab\Practicals\ADS Practicals>
```

ii) Postfix evaluation

Source Code:

```
import java.util.Stack;
public class PostEval {
 // Method to evaluate postfix expression
 public static int evaluatePostfix(String expr) {
 Stack<Integer> stack = new Stack<>();
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
// Loop through each character in the postfix expression
for (int i = 0; i < expr.length(); i++) {
char ch = expr.charAt(i);
// If the character is a digit, push it onto the stack
if (Character.isDigit(ch)) {
stack.push(ch - '0'); // Convert char to int
} else {
// Pop two elements from the stack and apply the operator
int b = stack.pop();
int a = stack.pop();
// Perform the operation based on the operator
switch (ch) {
case '+':
stack.push(a + b);
break;
case '-':
stack.push(a - b);
break;
case '*':
stack.push(a * b);
break;
case '/':
if (b != 0) {
stack.push(a / b); // Avoid division by zero
} else {
System.out.println("Division by zero error");
return -1; // Exit on division by zero error
}
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
 break;
 default:
 System.out.println("Invalid operator: " + ch);
 return -1; // Exit on invalid operator
 } } }
 // The final result will be at the top of the stack
 return stack.pop();
 }
 public static void main(String[] args) {
System.out.println("Roll no. 15");
 java.util.Scanner scanner = new java.util.Scanner(System.in);
 System.out.print("Enter Postfix Expression: ");
 String expr = scanner.next();
 int result = evaluatePostfix(expr);
 if (result != -1) {
 System.out.println("Result: " + result);
 }
 scanner.close();
 }  }
```

Output :

```
Roll no. 15
Enter Postfix Expression: 97+8-
Result: 8
PS D:\Documents\MCA\ADS Lab\Practicals\ADS Practicals>
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

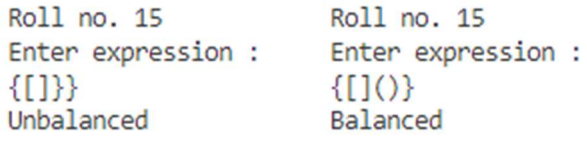
iii) Balancing of Parenthesis

Source Code:

```
import java.util.Scanner;
import java.util.Stack;
public class BalancedExpression {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
System.out.println("\nRoll no. 15 \nEnter expression :");
String expression = scanner.nextLine();
if (isBalanced(expression)) {
System.out.println("Balanced");
} else {
System.out.println("Unbalanced");
}
scanner.close();
}
private static boolean isBalanced(String expression) {
Stack<Character> stack = new Stack<>();
for (char ch : expression.toCharArray()) {
if (ch == '{' || ch == '[' || ch == '(') {
stack.push(ch);
} else {
switch (ch) {
case ')':
if (stack.isEmpty() || stack.pop() != '(') {
return false;
}
```

```
break;
case ']':
if (stack.isEmpty() || stack.pop() != '[') {
return false;
}
break;
case '}':
if (stack.isEmpty() || stack.pop() != '{') {
return false;
}
break;
default:
System.out.println("Enter the correct choice");
return false;
} } }
return stack.isEmpty();
} }
```

Output :



Name: Vishwajit Gadale
Roll No: 15
FYMCA

PRACTICAL NO:- 05

Aim:- Implementation of all types of linked lists.

i)Singly Linked Lists
Source Code:

```
import java.util.Scanner;
class SinglyLinkedList {
 // Node class for the linked list
 class Node {
 int data;
 Node next;
 public Node(int data) {
 this.data = data;
 this.next = null;
 } }
 private Node start = null;
 // Insert at the beginning
 public void insertAtBeg(int x) {
 Node newNode = new Node(x);
 if (start == null) {
 start = newNode;
 } else {
 newNode.next = start;
 start = newNode;
 } }
 // Insert at the end
 public void insertAtEnd(int x) {
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
Node newNode = new Node(x);
if (start == null) {
start = newNode;
} else {
Node p = start;
while (p.next != null) {
p = p.next;
}
p.next = newNode;
} }
// Insert at specific position
public void insertAtPos(int x, int pos) {
Node newNode = new Node(x);
if (start == null) {
System.out.println("List is empty.");
return;
}
Node p = start;
int count = 1;
// Traverse to position
while (p != null && count < pos - 1) {
p = p.next;
count++;
}
if (p == null || count < pos - 1) {
System.out.println("Invalid position.");
} else {
newNode.next = p.next;
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
p.next = newNode;
} }
// Search for a value
public void searchPos(int value) {
if (start == null) {
System.out.println("List is empty.");
return;
}
Node p = start;
int count = 1;
while (p != null) {
if (p.data == value) {
System.out.println("Value found at position " + count + ".");
return;
}
p = p.next;
count++;
}
System.out.println("Value not found.");
}
// Delete a node at a specific position
public void del(int pos) {
if (start == null) {
System.out.println("List is empty.");
return;
}
if (pos == 1) {
start = start.next;
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
  return;
  }
  Node p = start;
  int count = 1;
  while (p != null && count < pos - 1) {
  p = p.next;
  count++;
  }
  if (p == null || p.next == null) {
  System.out.println("Invalid position.");
  } else {
  p.next = p.next.next;
  }  }
  // Sort the list
  public void sort() {
  if (start == null) {
  System.out.println("List is empty.");
  return;
  }
  Node ptr = start;
  while (ptr != null) {
  Node p = ptr.next;
  while (p != null) {
  if (ptr.data > p.data) {
  int temp = ptr.data;
  ptr.data = p.data;
  p.data = temp;
  }
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
p = p.next;
}
ptr = ptr.next;
} }
// Reverse the list
public void reverse() {
if (start == null) {
System.out.println("List is empty.");
return;
}
if (start.next == null) {
System.out.println("Only one element in the list.");
return;
}
Node prev = null;
Node current = start;
Node next = null;
while (current != null) {
next = current.next;
current.next = prev;
prev = current;
current = next;
}
start = prev;
System.out.println("List reversed.");
}
// Display the list
public void display() {
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
  if (start == null) {
  System.out.println("List is empty.");
  return;
  }
  Node p = start;
  System.out.println("\nSingly Linked List:");
  while (p != null) {
  System.out.print(p.data + " -> ");
  p = p.next;
  }
  System.out.println();
  } }
 public class SinglyLinkedLists {
  public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);
  SinglyLinkedList list = new SinglyLinkedList();
 System.out.println("Roll no.15");
  int ch, x, pos;
  while (true) {
  System.out.println("\n1. Insert at beginning");
  System.out.println("2. Insert at end");
  System.out.println("3. Insert at position");
  System.out.println("4. Delete");
  System.out.println("5. Search");
  System.out.println("6. Display");
  System.out.println("7. Sort");
  System.out.println("8. Reverse");
  System.out.println("9. Exit");
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
System.out.print("Enter your choice: ");
ch = sc.nextInt();
switch (ch) {
case 1:
System.out.print("Enter the value: ");
x = sc.nextInt();
list.insertAtBeg(x);
list.display();
break;
case 2:
System.out.print("Enter the value: ");
x = sc.nextInt();
list.insertAtEnd(x);
list.display();
break;
case 3:
System.out.print("Enter the value: ");
x = sc.nextInt();
System.out.print("Enter the position: ");
pos = sc.nextInt();
list.insertAtPos(x, pos);
list.display();
break;
case 4:
System.out.print("Enter the position to delete: ");
pos = sc.nextInt();
list.del(pos);
list.display();
```

```
  break;
  case 5:
  System.out.print("Enter the value to search: ");
  x = sc.nextInt();
  list.searchPos(x);
  break;
  case 6:
  list.display();
  break;
  case 7:
  list.sort();
  list.display();
  break;
  case 8:
  list.reverse();
  list.display();
  break;
  case 9:
  sc.close();
  System.out.println("Exiting program.");
  return;
  default:
  System.out.println("Invalid choice.");
  }
  }
 }
 }
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

Output :

1. Insert at beginning:

```
Roll no.15

1. Insert at beginning
2. Insert at end
3. Insert at position
4. Delete
5. Search
6. Display
7. Sort
8. Reverse
9. Exit
Enter your choice: 1
Enter the value: 67

Singly Linked List:
67 ->
```

```
1. Insert at beginning
2. Insert at end
3. Insert at position
4. Delete
5. Search
6. Display
7. Sort
8. Reverse
9. Exit
Enter your choice: 1
Enter the value: 43

Singly Linked List:
43 -> 67 ->
```

```
1. Insert at beginning
2. Insert at end
3. Insert at position
4. Delete
5. Search
6. Display
7. Sort
8. Reverse
9. Exit
Enter your choice: 1
Enter the value: 83

Singly Linked List:
83 -> 43 -> 67 ->
```

2. Insert at end :

```
1. Insert at beginning
2. Insert at end
3. Insert at position
4. Delete
5. Search
6. Display
7. Sort
8. Reverse
9. Exit
Enter your choice: 2
Enter the value: 42

Singly Linked List:
83 -> 43 -> 67 -> 42 ->
```

```
1. Insert at beginning
2. Insert at end
3. Insert at position
4. Delete
5. Search
6. Display
7. Sort
8. Reverse
9. Exit
Enter your choice: 2
Enter the value: 64

Singly Linked List:
83 -> 43 -> 67 -> 42 -> 64 ->
```

```
1. Insert at beginning
2. Insert at end
3. Insert at position
4. Delete
5. Search
6. Display
7. Sort
8. Reverse
9. Exit
Enter your choice: 2
Enter the value: 78

Singly Linked List:
83 -> 43 -> 67 -> 42 -> 64 -> 78 ->
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

3. Insert at position :

```
1. Insert at beginning
2. Insert at end
3. Insert at position
4. Delete
5. Search
6. Display
7. Sort
8. Reverse
9. Exit
Enter your choice: 3
Enter the value: 44
Enter the position: 2

Singly Linked List:
83 -> 44 -> 43 -> 67 -> 42 -> 64 -> 78 ->
```

4.Delete :

```
1. Insert at beginning
2. Insert at end
3. Insert at position
4. Delete
5. Search
6. Display
7. Sort
8. Reverse
9. Exit
Enter your choice: 4
Enter the position to delete: 1

Singly Linked List:
44 -> 43 -> 67 -> 42 -> 64 -> 78 ->
```

5.Search

```
1. Insert at beginning
2. Insert at end
3. Insert at position
4. Delete
5. Search
6. Display
7. Sort
8. Reverse
9. Exit
Enter your choice: 5
Enter the value to search: 64
Value found at position 5.
```

6.Display

```
1. Insert at beginning
2. Insert at end
3. Insert at position
4. Delete
5. Search
6. Display
7. Sort
8. Reverse
9. Exit
Enter your choice: 6

Singly Linked List:
44 -> 43 -> 67 -> 42 -> 64 -> 78 ->
```

7. Sort

```
1. Insert at beginning
2. Insert at end
3. Insert at position
4. Delete
5. Search
6. Display
7. Sort
8. Reverse
9. Exit
Enter your choice: 7

Singly Linked List:
42 -> 43 -> 44 -> 64 -> 67 -> 78 ->
```

8.Reverse

```
1. Insert at beginning
2. Insert at end
3. Insert at position
4. Delete
5. Search
6. Display
7. Sort
8. Reverse
9. Exit
Enter your choice: 8
List reversed.

Singly Linked List:
78 -> 67 -> 64 -> 44 -> 43 -> 42 ->
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

9.Exit

```
1. Insert at beginning
2. Insert at end
3. Insert at position
4. Delete
5. Search
6. Display
7. Sort
8. Reverse
9. Exit
Enter your choice: 9
Exiting program.
```

ii) Circular Linked List

Source Code:

```
import java.util.Scanner;
class SinglyCircularLinkedList {
 private Node last = null;
 private int count = 0;
 // Node class for the circular linked list
 class Node {
 int data;
 Node next;
 Node(int data) {
 this.data = data;
 this.next = null;
 } }
 // Create the list with a single node
 public void create(int x) {
 Node tmp = new Node(x);
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
if (last == null) {
last = tmp;
tmp.next = last; // Points to itself, forming a circle
} else {
tmp.next = last.next;
last.next = tmp;
last = tmp; // Update the last node to the new node
} }
// Add element at the beginning
public void addAtBegin(int x) {
if (last == null) {
System.out.println("List is empty.");
return;
}
Node tmp = new Node(x);
tmp.next = last.next;
last.next = tmp;
}
// Add element after a given position
public void addAfter(int x, int pos) {
if (last == null) {
System.out.println("List is empty.");
return;
}
Node p = last.next;
for (int i = 0; i < pos - 1; i++) {
p = p.next;
if (p == last.next) {
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
System.out.println("Position does not exist.");
return;
} }
Node tmp = new Node(x);
tmp.next = p.next;
p.next = tmp;
if (p == last) {
last = tmp; // Update last if added at the end
} }
// Delete a node with the given value
public void del(int x) {
if (last == null) {
System.out.println("List is empty.");
return;
}
Node p = last.next;
if (last.next == last && last.data == x) {
last = null; // Single node case
return;
}
if (p.data == x) { // If first node needs to be deleted
last.next = p.next;
return;
}
while (p.next != last) {
if (p.next.data == x) {
p.next = p.next.next;
if (p.next == last) {
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
last = p; // Update last if last node is deleted
}
return;
}
p = p.next;
}
if (p.next.data == x) { // Deleting the last node
p.next = last.next;
last = p;
}
System.out.println("Element not found.");
}
// Search for an element and print its position
public void search(int x) {
int pos = 1;
if (last == null) {
System.out.println("List is empty.");
return;
}
Node p = last.next;
while (p != last) {
if (p.data == x) {
System.out.println("Element found at position " + pos + ".");
return;
}
p = p.next;
pos++;
}
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
if (last.data == x) {
System.out.println("Element found at position " + pos + ".");
} else {
System.out.println("Item not found.");
}  }
// Sort the list using bubble sort
public void sort() {
if (last == null) {
System.out.println("List is empty.");
return;
}
Node p = last.next;
Node ptr = null;
int temp;
while (p != last) {
ptr = p.next;
while (ptr != last.next) {
if (p.data > ptr.data) {
temp = p.data;
p.data = ptr.data;
ptr.data = temp;
}
ptr = ptr.next;
}
p = p.next;
}  }
// Count the number of elements in the list
public void count() {
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
  if (last == null) {
  System.out.println("List is empty.");
  return;
  }
  Node p = last.next;
  int count = 0;
  do {
  count++;
  p = p.next;
  } while (p != last.next);
  System.out.println("Number of elements: " + count);
  }
  // Display the list
  public void display() {
  if (last == null) {
  System.out.println("List is empty.");
  return;
  }
  Node p = last.next;
  System.out.println("\nSingly Circular Linked List:");
  do {
  System.out.print(p.data + " -> ");
  p = p.next;
  } while (p != last.next);
  System.out.println("(head)");
  }  }
 public class CircularLinkedLists {
  public static void main(String[] args) {
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
Scanner sc = new Scanner(System.in);
SinglyCircularLinkedList list = new SinglyCircularLinkedList();
int ch, x, pos;
System.out.println("Roll no.15");
while (true) {
System.out.println("1. Create a list");
System.out.println("2. Add at begin");
System.out.println("3. Add after");
System.out.println("4. Search");
System.out.println("5. Sort");
System.out.println("6. Count");
System.out.println("7. Display");
System.out.println("8. Delete");
System.out.println("9. Exit");
System.out.print("Enter the choice: ");
ch = sc.nextInt();
switch (ch) {
case 1:
System.out.print("Enter the value: ");
x = sc.nextInt();
list.create(x);
list.display();
break;
case 2:
System.out.print("Enter the value: ");
x = sc.nextInt();
list.addAtBegin(x);
list.display();
```

```
  break;
  case 3:
  System.out.print("Enter the position: ");
  pos = sc.nextInt();
  System.out.print("Enter the value: ");
  x = sc.nextInt();
  list.addAfter(x, pos);
  list.display();
  break;
  case 4:
  System.out.print("Enter element to be searched: ");
  x = sc.nextInt();
  list.search(x);
  break;
  case 5:
  System.out.println("Before sorting:");
  list.display();
  list.sort();
  System.out.println("After sorting:");
  list.display();
  break;
  case 6:
  list.count();
  break;
  case 7:
  list.display();
  break;
  case 8:
```

```
System.out.print("Enter the element to delete: ");
x = sc.nextInt();
list.del(x);
list.display();
break;
case 9:
sc.close();
return;
default:
System.out.println("Wrong choice.");
} } } }
```

Output :

1. Create a List :

```
Roll no.15
1. Create a list
2. Add at begin
3. Add after
4. Search
5. Sort
6. Count
7. Display
8. Delete
9. Exit
Enter the choice: 1
Enter the value: 13

Singly Circular Linked List:
13 -> (head)
```

```
1. Create a list
2. Add at begin
3. Add after
4. Search
5. Sort
6. Count
7. Display
8. Delete
9. Exit
Enter the choice: 1
Enter the value: 15

Singly Circular Linked List:
13 -> 43 -> 15 -> (head)
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

2. Add at Begin :

```
1. Create a list
2. Add at begin
3. Add after
4. Search
5. Sort
6. Count
7. Display
8. Delete
9. Exit
Enter the choice: 2
Enter the value: 34

Singly Circular Linked List:
34 -> 15 -> 13 -> (head)
```

3.Add After :

```
1. Create a list
2. Add at begin
3. Add after
4. Search
5. Sort
6. Count
7. Display
8. Delete
9. Exit
Enter the choice: 3
Enter the position: 2
Enter the value: 45

Singly Circular Linked List:
34 -> 15 -> 45 -> 13 -> (head)
```

4.Search :

```
1. Create a list
2. Add at begin
3. Add after
4. Search
5. Sort
6. Count
7. Display
8. Delete
9. Exit
Enter the choice: 4
Enter element to be searched: 15
Element found at position 2.
```

5.Sort :

```
1. Create a list
2. Add at begin
3. Add after
4. Search
5. Sort
6. Count
7. Display
8. Delete
9. Exit
Enter the choice: 5
Before sorting:

Singly Circular Linked List:
13 -> 15 -> 34 -> 45 -> (head)
After sorting:
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

6.Count :

```
1. Create a list
2. Add at begin
3. Add after
4. Search
5. Sort
6. Count
7. Display
8. Delete
9. Exit
Enter the choice: 6
Number of elements: 4
```

7.Display :

```
1. Create a list
2. Add at begin
3. Add after
4. Search
5. Sort
6. Count
7. Display
8. Delete
9. Exit
Enter the choice: 7

Singly Circular Linked List:
13 -> 15 -> 34 -> 45 -> (head)
```

8.Delete :

```
1. Create a list
2. Add at begin
3. Add after
4. Search
5. Sort
6. Count
7. Display
8. Delete
9. Exit
Enter the choice: 8
Enter the element to delete: 34

Singly Circular Linked List:
45 -> 13 -> 15 -> (head)
```

9.Exit :

```
1. Create a list
2. Add at begin
3. Add after
4. Search
5. Sort
6. Count
7. Display
8. Delete
9. Exit
Enter the choice: 9
PS D:\Documents\MCA\ADS Lab\Practicals\ADS Practicals>
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

iii) Doubly Linked Lists

Source Code:

```
import java.util.Scanner;
public class DoublyLinkedList {
 static class Node {
 int data;
 Node next;
 Node prev;
 Node(int data) {
 this.data = data;
 this.next = null;
 this.prev = null;
 }  }
 private Node start = null;
 // Create a list
 public void create(int x) {
 Node tmp = new Node(x);
 if (start == null) {
 start = tmp;
 } else {
 Node p = start;
 while (p.next != null) {
 p = p.next;
 }
 p.next = tmp;
 tmp.prev = p;
 }  }
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
// Add at the beginning
public void addAtBegin(int x) {
if (start == null) {
System.out.println("List is empty.");
} else {
Node tmp = new Node(x);
tmp.next = start;
start.prev = tmp;
start = tmp;
}  }
// Add after a given position
public void addAfter(int x, int pos) {
if (start == null) {
System.out.println("List is empty.");
} else {
Node p = start;
for (int i = 1; i < pos; i++) {
if (p == null) {
System.out.println("Position does not exist.");
return;
}
p = p.next;
}
Node tmp = new Node(x);
tmp.next = p.next;
if (p.next != null) {
p.next.prev = tmp;
}
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
p.next = tmp;
tmp.prev = p;
} }
// Delete an element
public void delete(int x) {
if (start == null) {
System.out.println("List is empty.");
return;
}
// Delete first element
if (start.data == x) {
Node tmp = start;
start = start.next;
if (start != null) start.prev = null;
tmp = null;
return;
}
// Delete middle or last element
Node p = start;
while (p != null && p.next != null) {
if (p.next.data == x) {
Node tmp = p.next;
p.next = tmp.next;
if (tmp.next != null) {
tmp.next.prev = p;
}
tmp = null;
return;
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
  }
  p = p.next;
  }
  // If we reach here, the element was not found
  if (p != null && p.data == x) {
  Node tmp = p;
  if (p.prev != null) p.prev.next = null;
  tmp = null;
  }  }
  // Reverse the list
  public void reverse() {
  Node p1 = start;
  Node p2 = (p1 != null) ? p1.next : null;
  p1.next = null;
  if (p1 != null) p1.prev = p2;
  while (p2 != null) {
  p2.prev = p2.next;
  p2.next = p1;
  p1 = p2;
  p2 = p2.prev;
  }
  start = p1;
  System.out.println("List reversed.");
  }
  // Count elements in the list
  public void count() {
  Node p = start;
  int cnt = 0;
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
while (p != null) {
cnt++;
p = p.next;
}
System.out.println("Number of elements are " + cnt + ".");
}
// Search for an element
public void search() {
Scanner sc = new Scanner(System.in);
System.out.print("Enter the element to be searched: ");
int value = sc.nextInt();
if (start == null) {
System.out.println("List is empty.");
return;
}
Node p = start;
int count = 0;
while (p != null) {
count++;
if (p.data == value) {
System.out.println("Element found at position " + count + ".");
return;
}
p = p.next;
}
System.out.println("Element not found.");
}
// Sort the list
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
public void sort() {
if (start == null) {
System.out.println("List is empty.");
return;
}
Node ptr = start;
while (ptr != null) {
Node p = ptr.next;
while (p != null) {
if (ptr.data > p.data) {
int temp = ptr.data;
ptr.data = p.data;
p.data = temp;
}
p = p.next;
}
ptr = ptr.next;
} }
// Display the list
public void display() {
if (start == null) {
System.out.println("List is empty.");
return;
}
Node p = start;
System.out.print("\nDoubly Linked List: ");
while (p != null) {
System.out.print(p.data + " <-> ");
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
  p = p.next;
  }
  System.out.println("null\n");
  }
  public static void main(String[] args) {
  DoublyLinkedList d = new DoublyLinkedList();
 System.out.println("Roll no.15");
  Scanner sc = new Scanner(System.in);
  int x, pos, ch;
  while (true) {
  System.out.println("1. Create a list");
  System.out.println("2. Add at begin");
  System.out.println("3. Add after");
  System.out.println("4. Search");
  System.out.println("5. Reverse");
  System.out.println("6. Count");
  System.out.println("7. Sort");
  System.out.println("8. Display");
  System.out.println("9. Delete");
  System.out.println("10. Exit");
  System.out.print("Enter your choice: ");
  ch = sc.nextInt();
  switch (ch) {
  case 1:
  System.out.print("Enter the value: ");
  x = sc.nextInt();
  d.create(x);
  d.display();
```

```
break;
case 2:
System.out.print("Enter the value: ");
x = sc.nextInt();
d.addAtBegin(x);
d.display();
break;
case 3:
System.out.print("Enter the position: ");
pos = sc.nextInt();
System.out.print("Enter the value: ");
x = sc.nextInt();
d.addAfter(x, pos);
d.display();
break;
case 4:
d.search();
d.display();
break;
case 5:
d.reverse();
d.display();
break;
case 6:
d.count();
d.display();
break;
case 7:
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
  System.out.println("Before sorting:");
  d.display();
  d.sort();
  System.out.println("After sorting:");
  d.display();
  break;
  case 8:
  d.display();
  break;
  case 9:
  System.out.print("Enter the element to be deleted: ");
  x = sc.nextInt();
  d.delete(x);
  d.display();
  break;
  case 10:
  sc.close();
  System.exit(0);
  break;
  default:
  System.out.println("Invalid choice");
  }
  }
 }
 }
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

Output :

1.Create a List :

```
Roll no.15
1. Create a list
2. Add at begin
3. Add after
4. Search
5. Reverse
6. Count
7. Sort
8. Display
9. Delete
10. Exit
Enter your choice: 1
Enter the value: 15

Doubly Linked List: 15 <-> null
```

```
1. Create a list
2. Add at begin
3. Add after
4. Search
5. Reverse
6. Count
7. Sort
8. Display
9. Delete
10. Exit
Enter your choice: 1
Enter the value: 45

Doubly Linked List: 15 <-> 45 <-> null
```

2.Add at Begin :

```
1. Create a list
2. Add at begin
3. Add after
4. Search
5. Reverse
6. Count
7. Sort
8. Display
9. Delete
10. Exit
Enter your choice: 2
Enter the value: 65

Doubly Linked List: 65 <-> 15 <-> 45 <-> null
```

3.Add After :

```
1. Create a list
2. Add at begin
3. Add after
4. Search
5. Reverse
6. Count
7. Sort
8. Display
9. Delete
10. Exit
Enter your choice: 3
Enter the position: 2
Enter the value: 78

Doubly Linked List: 65 <-> 15 <-> 78 <-> 45 <-> null
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

3. Search :

```
1. Create a list
2. Add at begin
3. Add after
4. Search
5. Reverse
6. Count
7. Sort
8. Display
9. Delete
10. Exit
Enter your choice: 4
Enter the element to be searched: 78
Element found at position 3.

Doubly Linked List: 65 <-> 15 <-> 78 <-> 45 <-> null
```

5. Reverse :

```
1. Create a list
2. Add at begin
3. Add after
4. Search
5. Reverse
6. Count
7. Sort
8. Display
9. Delete
10. Exit
Enter your choice: 5
List reversed.

Doubly Linked List: 45 <-> 78 <-> 15 <-> 65 <-> null
```

6.Count :

```
1. Create a list
2. Add at begin
3. Add after
4. Search
5. Reverse
6. Count
7. Sort
8. Display
9. Delete
10. Exit
Enter your choice: 6
Number of elements are 4.

Doubly Linked List: 45 <-> 78 <-> 15 <-> 65 <-> null
```

7. Sort :

```
1. Create a list
2. Add at begin
3. Add after
4. Search
5. Reverse
6. Count
7. Sort
8. Display
9. Delete
10. Exit
Enter your choice: 7
Before sorting:

Doubly Linked List: 45 <-> 78 <-> 15 <-> 65 <-> null

After sorting:

Doubly Linked List: 15 <-> 45 <-> 65 <-> 78 <-> null
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

8. Display :

```
1. Create a list
2. Add at begin
3. Add after
4. Search
5. Reverse
6. Count
7. Sort
8. Display
9. Delete
10. Exit
Enter your choice: 8

Doubly Linked List: 15 <-> 45 <-> 65 <-> 78 <-> null
```

9.Delete :

```
1. Create a list
2. Add at begin
3. Add after
4. Search
5. Reverse
6. Count
7. Sort
8. Display
9. Delete
10. Exit
Enter your choice: 9
Enter the element to be deleted: 65

Doubly Linked List: 15 <-> 45 <-> 78 <-> null
```

10. Exit :

```
1. Create a list
2. Add at begin
3. Add after
4. Search
5. Reverse
6. Count
7. Sort
8. Display
9. Delete
10. Exit
Enter your choice: 10
PS D:\Documents\MCA\ADS Lab\Practicals\ADS Practicals>
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

iv) Polynomial Addition

Source Code:

```
public class PolynomialAddition {

    // Method to find the maximum of two integers
    public static int max(int m, int n) {
    return (m > n) ? m : n;
    }
    // Method to add two polynomials
    public static int[] add(int[] A, int[] B, int m, int n) {
    int size = max(m, n);
    int[] sum = new int[size];

    // Copying elements from A[] to sum[]
    for (int i = 0; i < m; i++) {
    sum[i] = A[i];
    }

    // Adding elements from B[] to sum[]
    for (int i = 0; i < n; i++) {
    sum[i] += B[i];
    }

    return sum;
    }
    // Method to print the polynomial
    public static void printPoly(int[] poly, int n) {
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
    for (int i = 0; i < n; i++) {
    System.out.print(poly[i]);
    if (i != 0) {
    System.out.print("x^" + i);
    }
    if (i != n - 1) {
    System.out.print(" + ");
    }
    }
    System.out.println();
    }
    public static void main(String[] args) {
    // Student details
    System.out.println("Roll No:15");

    System.out.println("\nPOLYNOMIAL ADDITION\n");

    // First polynomial A = 1 + 0x + 2x^2 + 4x^3
    int[] A = {1, 0, 2, 4};

    // Second polynomial B = 3 + 5x + 7x^2
    int[] B = {3, 5, 7};

    // Sizes of A[] and B[]
    int m = A.length;
    int n = B.length;
    // Printing the first polynomial
    System.out.println("First polynomial is ");
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
    printPoly(A, m);
    // Printing the second polynomial
    System.out.println("Second polynomial is ");
    printPoly(B, n);
    // Adding polynomials A[] and B[] and getting the sum polynomial
    int[] sum = add(A, B, m, n);
    int size = max(m, n);

    // Printing the sum of polynomials
    System.out.println("\nsum polynomial is ");
    printPoly(sum, size);
    }
   }
```

Output :

```
Roll No:15

POLYNOMIAL ADDITION

First polynomial is
1 + 0x^1 + 2x^2 + 4x^3
Second polynomial is
3 + 5x^1 + 7x^2

sum polynomial is
4 + 5x^1 + 9x^2 + 4x^3
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

PRACTICAL NO:- 06

Aim:-Demonstrate application of linked list

i)Stack

Source Code:

```
import java.util.Scanner;
// Node class representing each element in the stack
class Node {
    int data;
    Node next;
    // Constructor for Node
    Node(int value) {
        data = value;
        next = null;
    }
}
// Stack class implementing stack operations using a linked list
class Stack {
    private Node top;
    // Constructor for Stack
    public Stack() {
        top = null;
    }
    // Push operation - adds an element to the top of the stack
    public void push(int value) {
        Node newNode = new Node(value);
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
        newNode.next = top;
        top = newNode;
        System.out.println(value + " pushed to stack");
    }
    // Pop operation - removes the top element from the stack
    public int pop() {
        if (top == null) {
            System.out.println("Stack is empty");
            return -1;
        }
        int value = top.data;
        top = top.next; // Move the top pointer to the next node
        return value;
    }
    // Peek operation - returns the top element without removing it
    public int peek() {
        if (top == null) {
            System.out.println("Stack is empty");
            return -1;
        }
        return top.data;
    }
    // Display operation - displays the contents of the stack
    public void display() {
        if (top == null) {
            System.out.println("Stack is empty");
            return;
        }
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
        Node current = top;
        System.out.print("Stack contents: ");
        while (current != null) {
            System.out.print(current.data + " ");
            current = current.next;
        }
        System.out.println();
    }
}
// Main class to interact with the stack
public class Main {
    public static void main(String[] args) {
        System.out.println("Roll No. 15");
        Stack stack = new Stack();
        Scanner scanner = new Scanner(System.in);
        int choice, value;
        do {
            // Display menu options
            System.out.println("\nStack Operations Menu:");
            System.out.println("1. Push");
            System.out.println("2. Pop");
            System.out.println("3. Peek");
            System.out.println("4. Display");
            System.out.println("5. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();
            switch (choice) {
                case 1: // Push operation
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
                System.out.print("Enter value to push: ");
                value = scanner.nextInt();
                stack.push(value);
                break;
            case 2: // Pop operation
                value = stack.pop();
                if (value != -1) {
                    System.out.println(value + " popped from stack");
                }
                break;
            case 3: // Peek operation
                value = stack.peek();
                if (value != -1) {
                    System.out.println("Top element is: " + value);
                }
                break;
            case 4: // Display operation
                stack.display();
                break;
            case 5: // Exit
                System.out.println("Exiting...");
                break;
            default: // Invalid choice
                System.out.println("Invalid choice. Please try again.");
        }
    } while (choice != 5);
    scanner.close();
} }
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

Output :

```
Stack Operations Menu:
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your choice: 1
Enter value to push: 15
15 pushed to stack

Stack Operations Menu:
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your choice: 1
Enter value to push: 45
45 pushed to stack
```

```
Stack Operations Menu:
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your choice: 4
Stack contents: 45 15

Stack Operations Menu:
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your choice: 2
45 popped from stack
```

```
Stack Operations Menu:
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your choice: 3
Top element is: 15

Stack Operations Menu:
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your choice: 5
Exiting...
```

ii) Ordinary Queue

Source Code:

```
import java.util.Scanner;

// Node class representing each element in the queue
class Node {
    int data;
    Node next;

    // Constructor for Node
    Node(int value) {
        data = value;
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
        next = null;
    }
}

// Queue class implementing queue operations using a linked list
class Queue {
    private Node front; // Points to the front of the queue
    private Node rear;  // Points to the rear of the queue

    // Constructor for Queue
    public Queue() {
        front = rear = null;
    }

    // Enqueue operation - adds an element to the end of the queue
    public void enqueue(int value) {
        Node newNode = new Node(value);
        if (rear == null) { // If the queue is empty
            front = rear = newNode;
        } else {
            rear.next = newNode;
            rear = newNode;
        }
        System.out.println(value + " enqueued to queue");
    }

    // Dequeue operation - removes an element from the front of the queue
    public int dequeue() {
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
        if (front == null) { // If the queue is empty
            System.out.println("Queue is empty");
            return -1;
        }
        int value = front.data;
        front = front.next; // Move the front pointer to the next node
        if (front == null) { // If the queue becomes empty
            rear = null;
        }
        return value;
    }

    // Peek operation - returns the front element without removing it
    public int peek() {
        if (front == null) { // If the queue is empty
            System.out.println("Queue is empty");
            return -1;
        }
        return front.data;
    }

    // Display operation - shows all elements in the queue
    public void display() {
        if (front == null) { // If the queue is empty
            System.out.println("Queue is empty");
            return;
        }
        Node current = front;
```

```
        System.out.print("Queue contents: ");
        while (current != null) {
            System.out.print(current.data + " ");
            current = current.next;
        }
        System.out.println();
    }
}

// Main class to test the queue implementation
public class OrdinaryQueue {
    public static void main(String[] args) {
        Queue queue = new Queue();
        System.out.println("Roll No. 15");
        Scanner scanner = new Scanner(System.in);
        int choice, value;

        do {
            // Display menu options
            System.out.println("\nQueue Operations Menu:");
            System.out.println("1. Enqueue");
            System.out.println("2. Dequeue");
            System.out.println("3. Display");
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();

            switch (choice) {
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
            case 1: // Enqueue operation
                System.out.print("Enter value to enqueue: ");
                value = scanner.nextInt();
                queue.enqueue(value);
                break;

            case 2: // Dequeue operation
                value = queue.dequeue();
                if (value != -1) {
                    System.out.println(value + " dequeued from queue");
                }
                break;

            case 3: // Display operation
                queue.display();
                break;

            case 4: // Exit
                System.out.println("Exiting...");
                break;

            default: // Invalid choice
                System.out.println("Invalid choice. Please try again.");
        }
    } while (choice != 4);

    scanner.close();
} }
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

Output :

```
Roll No. 15

Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter value to enqueue: 15
15 enqueued to queue

Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter value to enqueue: 45
45 enqueued to queue
```

```
Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 2
15 dequeued from queue

Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 4
Exiting...
```

iii) Priority Queue :

Source Code:

```
import java.util.Scanner;
class Node {
int data;
int priority;
Node next;
Node(int data, int priority) {
this.data = data;
this.priority = priority;
this.next = null;
}
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
}
class PriorityQueue {
private Node front;
public PriorityQueue() {
front = null;
}
public void enqueue(int value, int priority) {
Node newNode = new Node(value, priority);
if (front == null || priority < front.priority) {
newNode.next = front;
front = newNode;
} else {
Node current = front;
while (current.next != null && current.next.priority <= priority) {
current = current.next;
}
newNode.next = current.next;
current.next = newNode;
}
System.out.println(value + " enqueued to queue with priority " + priority);
}
public int dequeue() {
if (front == null) {
System.out.println("Queue is empty");
return -1;
}
Node temp = front;
int value = front.data;
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
front = front.next;
temp = null; // Help with garbage collection
return value;
}
public int peek() {
if (front == null) {
System.out.println("Queue is empty");
return -1;
}
return front.data;
}
public void display() {
if (front == null) {
System.out.println("Queue is empty");
return;
}
Node current = front;
System.out.print("Priority Queue contents (data: priority): ");
while (current != null) {
System.out.print("(" + current.data + ": " + current.priority + ") ");
current = current.next;
}
System.out.println();
}
}
public class PriorityQueues {
public static void main(String[] args) {
PriorityQueue pq = new PriorityQueue();
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
System.out.println("Roll no. 15");
Scanner scanner = new Scanner(System.in);
int ch, value, priority;
do {
System.out.println("\nPriority Queue Operations Menu:");
System.out.println("1. Enqueue 2. Dequeue 3. Display 4. Exit");
System.out.print("Enter your choice: ");
ch = scanner.nextInt();
switch (ch) {
case 1:
System.out.print("Enter value to enqueue: ");
value = scanner.nextInt();
System.out.print("Enter priority (lower number = higher priority): ");
priority = scanner.nextInt();
pq.enqueue(value, priority);
break;
case 2:
value = pq.dequeue();
if (value != -1) {
System.out.println(value + " dequeued from queue");
}
break;
case 3:
pq.display();
break;
case 4:
System.exit(0);
default:
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
System.out.println("Invalid choice. Please try again.");

}

} while (ch != 4);

scanner.close();

}

}
```

Output :

1.Enqueue :

```
Roll no. 15

Priority Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter value to enqueue: 30
Enter priority (lower number = higher priority): 2
30 enqueued to queue with priority 2
```

```
Priority Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter value to enqueue: 50
Enter priority (lower number = higher priority): 1
50 enqueued to queue with priority 1
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

3.Display :

```
Priority Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
Priority Queue contents (data: priority): (50: 1) (40: 1) (30: 2)
```

2.Deuqeue :

```
Priority Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 2
50 dequeued from queue

Priority Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
Priority Queue contents (data: priority): (40: 1) (30: 2)
```

4.Exit :

```
Priority Queue Operations Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 4
Exiting...
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

iv) Double ended Queue :

Source Code:

```
import java.util.Scanner;
class Node {
int data;
Node next;
Node prev;
Node(int value) {
data = value;
next = null;
prev = null;
}
}
class Deque {
private Node front;
private Node rear;
public Deque() {
front = rear = null;
}
public void insertFront(int value) {
Node newNode = new Node(value);
newNode.next = front;
if (front != null) {
front.prev = newNode;
}
front = newNode;
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
if (rear == null) {
rear = newNode;
}
System.out.println(value + " inserted at front");
}
public void insertRear(int value) {
Node newNode = new Node(value);
newNode.next = null;
newNode.prev = rear;
if (rear != null) {
rear.next = newNode;
}
rear = newNode;
if (front == null) {
front = newNode;
}
System.out.println(value + " inserted at rear");
}
public int deleteFront() {
if (front == null) {
System.out.println("Deque is empty");
return -1;
}
int value = front.data;
front = front.next;
if (front != null) {
front.prev = null;
} else {
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
rear = null;
}
return value;
}
public int deleteRear() {
if (rear == null) {
System.out.println("Deque is empty");
return -1;
}
int value = rear.data;
rear = rear.prev;
if (rear != null) {
rear.next = null;
} else {
front = null;
}
return value;
}
public int getFront() {
if (front == null) {
System.out.println("Deque is empty");
return -1;
}
return front.data;
}
public int getRear() {
if (rear == null) {
System.out.println("Deque is empty");
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
return -1;
}
return rear.data;
}
public void display() {
if (front == null) {
System.out.println("Deque is empty");
return;
}
Node current = front;
System.out.print("Deque contents: ");
while (current != null) {
System.out.print(current.data + " ");
current = current.next;
}
System.out.println();
}
}
public class Main {
public static void main(String[] args) {
Deque dq = new Deque();
System.out.println("Roll no. 15");
Scanner scanner = new Scanner(System.in);
int ch, value;
System.out.println("\nDeque Operations Menu:");
System.out.println("1. Insert Front 2. Insert Rear 3. Delete Front");
System.out.println("4. Delete Rear 5. Display 6. Exit");
do {
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
System.out.print("Enter your choice: ");
ch = scanner.nextInt();
switch (ch) {
case 1:
System.out.print("Enter value to insert at front: ");
value = scanner.nextInt();
dq.insertFront(value);
break;
case 2:
System.out.print("Enter value to insert at rear: ");
value = scanner.nextInt();
dq.insertRear(value);
break;
case 3:
value = dq.deleteFront();
if (value != -1) {
System.out.println(value + " deleted from front");
}
break;
case 4:
value = dq.deleteRear();
if (value != -1) {
System.out.println(value + " deleted from rear");
}
break;
case 5:
dq.display();
break;
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
case 6:

System.exit(0);

default:

System.out.println("Invalid choice. Please try again.");

}

} while (ch != 6);

scanner.close();

}

}
```

Output :

```
Roll no. 15

Deque Operations Menu:
1. Insert Front 2. Insert Rear 3. Delete Front
4. Delete Rear 5. Display 6. Exit
Enter your choice: 1
Enter value to insert at front: 15
15 inserted at front
Enter your choice: 1
Enter value to insert at front: 65
65 inserted at front
Enter your choice: 5
Deque contents: 65 15
Enter your choice: 2
Enter value to insert at rear: 87
65 inserted at front
Enter your choice: 5
Deque contents: 65 15
Enter value to insert at rear: 87
65 inserted at front
Enter your choice: 5
Deque contents: 65 15
Enter your choice: 2
Enter value to insert at rear: 87
87 inserted at rear
87 inserted at rear
Enter your choice: 2
Enter value to insert at rear: 35
35 inserted at rear
Enter your choice: 5
Deque contents: 65 15 87 35
Enter your choice: 3
65 deleted from front
Enter your choice: 4
35 deleted from rear
Enter your choice: 5
Deque contents: 15 87
Enter your choice: 6
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

PRACTICAL NO:- 07

Aim:-Create and perform various operations on BST.

Source Code:

```
import java.util.Scanner;
class BinarySearchTree {
 // Node structure for the tree
 class Node {
 int data;
 Node left, right;
 public Node(int item) {
 data = item;
 left = right = null;
 }
 }
 private Node tree;
 public BinarySearchTree() {
 tree = null;
 }
 // Method to create a tree (insert node)
 public Node createTree(Node node, int item) {
 if (node == null) {
 node = new Node(item);
 } else {
 if (node.data > item) {
 node.left = createTree(node.left, item);
 } else {
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
node.right = createTree(node.right, item);
}
}
return node;
}
// Preorder traversal
public void preorder(Node node) {
if (node != null) {
System.out.print(" " + node.data);
preorder(node.left);
preorder(node.right);
}
}
// Inorder traversal
public void inorder(Node node) {
if (node != null) {
inorder(node.left);
System.out.print(" " + node.data);
inorder(node.right);
}
}
// Postorder traversal
public void postorder(Node node) {
if (node != null) {
postorder(node.left);
postorder(node.right);
System.out.print(" " + node.data);
}
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
 }
 // Count total nodes in the tree
 public int totalNodes(Node node) {
 if (node == null)
 return 0;
 return totalNodes(node.left) + totalNodes(node.right) + 1;
 }
 // Find the smallest node
 public void findSmallestNode(Node node) {
 if (node == null || node.left == null)
 System.out.println(node.data);
 else
 findSmallestNode(node.left);
 }
 // Find the largest node
 public void findLargestNode(Node node) {
 if (node == null || node.right == null)
 System.out.println(node.data);
 else
 findLargestNode(node.right);
 }
 // Main method to handle user input
 public static void main(String[] args) {
 BinarySearchTree obj = new BinarySearchTree();
 Scanner scanner = new Scanner(System.in);
 int choice, n, item;
System.out.println("Roll no.15");
 while (true) {
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
System.out.println("\nBinary search tree common operation");
System.out.println("1) Create Tree");
System.out.println("2) Traversal");
System.out.println("3) Total Nodes");
System.out.println("4) Insert Nodes");
System.out.println("5) Find Smallest Node");
System.out.println("6) Find Largest Node");
System.out.println("7) Exit");
System.out.print("Enter your choice: ");
choice = scanner.nextInt();
switch (choice) {
case 1:
System.out.print("\nCreating Tree----");
System.out.print("\nHow many nodes do you want to enter: ");
n = scanner.nextInt();
for (int i = 0; i < n; i++) {
System.out.print("Enter value: ");
item = scanner.nextInt();
obj.tree = obj.createTree(obj.tree, item);
}
break;
case 2:
System.out.println("\nInorder Traversal:");
obj.inorder(obj.tree);
System.out.println("\nPreorder Traversal:");
obj.preorder(obj.tree);
System.out.println("\nPostorder Traversal:");
obj.postorder(obj.tree);
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
 break;
 case 3:
 int total = obj.totalNodes(obj.tree);
 System.out.println("Total nodes: " + total);
 break;
 case 4:
 System.out.print("\nInsert node in a tree \nEnter value: ");
 item = scanner.nextInt();
 obj.tree = obj.createTree(obj.tree, item);
 System.out.println("\nItem is inserted.");
 break;
 case 5:
 System.out.println("\nSmallest node is:");
 obj.findSmallestNode(obj.tree);
 break;
 case 6:
 System.out.println("\nLargest node is:");
 obj.findLargestNode(obj.tree);
 break;
 case 7:
 System.out.println("Exiting program.");
 scanner.close();
 System.exit(0);
 default:
 System.out.println("Invalid choice, try again.");
 } }
 }
}
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

Output :

1.Create Tree :

```
Roll no.15

Binary search tree common operation
1) Create Tree
2) Traversal
3) Total Nodes
4) Insert Nodes
5) Find Smallest Node
6) Find Largest Node
7) Exit
Enter your choice: 1

Creating Tree----
How many nodes do you want to enter: 4
Enter value: 15
Enter value: 34
Enter value: 32
Enter value: 65
```

2.Traversal :

```
Binary search tree common operation
1) Create Tree
2) Traversal
3) Total Nodes
4) Insert Nodes
5) Find Smallest Node
6) Find Largest Node
7) Exit
Enter your choice: 2

Inorder Traversal:
 15 32 34 65
Preorder Traversal:
 15 34 32 65
Postorder Traversal:
 32 65 34 15
```

3.Total Nodes :

```
Binary search tree common operation
1) Create Tree
2) Traversal
3) Total Nodes
4) Insert Nodes
5) Find Smallest Node
6) Find Largest Node
7) Exit
Enter your choice: 3
Total nodes: 4
```

4. Insert nodes :

```
Binary search tree common operation
1) Create Tree
2) Traversal
3) Total Nodes
4) Insert Nodes
5) Find Smallest Node
6) Find Largest Node
7) Exit
Enter your choice: 4

Insert node in a tree
Enter value: 87

Item is inserted.
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

5. Find Smallest Node :

```
Binary search tree common operation
1) Create Tree
2) Traversal
3) Total Nodes
4) Insert Nodes
5) Find Smallest Node
6) Find Largest Node
7) Exit
Enter your choice: 5

Smallest node is:
15
```

6. Finding Largest Node :

```
Binary search tree common operation
1) Create Tree
2) Traversal
3) Total Nodes
4) Insert Nodes
5) Find Smallest Node
6) Find Largest Node
7) Exit
Enter your choice: 6

Largest node is:
87
Largest node is:
87
```

7 . Exit :

```
Binary search tree common operation
1) Create Tree
2) Traversal
3) Total Nodes
4) Insert Nodes
5) Find Smallest Node
6) Find Largest Node
7) Exit
Enter your choice: 7
Exiting program.
PS D:\Documents\MCA\ADS Lab\Practicals\ADS Practicals>
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

PRACTICAL NO:- 08

Aim:-Implementing Heap with different operations.

Source Code:

```
import java.util.Scanner;
public class MaxHeap {
 // Method to perform max heapify
 public static void maxHeapify(int[] a, int i, int n) {
 int j, temp;
 temp = a[i];
 j = 2 * i;
 // Perform max-heapify by comparing the node with its children
 while (j <= n) {
 if (j < n && a[j + 1] > a[j]) {
 j = j + 1; // If right child is larger, select right child
 }
 if (temp > a[j]) {
 break; // If the node is larger than or equal to the largest child, stop
 } else if (temp <= a[j]) {
 a[j / 2] = a[j]; // Swap the node with the largest child
 j = 2 * j; // Move down the tree
 }
 }
 a[j / 2] = temp; // Place the original value at the correct position
 }
 // Method to build the max heap from an unsorted array
 public static void buildMaxHeap(int[] a, int n) {
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
 for (int i = n / 2; i >= 1; i--) {
 maxHeapify(a, i, n); // Call maxHeapify on all non-leaf nodes
 }
 }
 public static void main(String[] args) {
 Scanner scanner = new Scanner(System.in);
System.out.println("Roll no: 15");
 // Input the number of elements
 System.out.println("Enter number of elements in the array:");
 int n = scanner.nextInt();
 // Initialize the array
 int[] a = new int[21]; // Size 21 to handle 1-based indexing
 // Input the elements into the array
 for (int i = 1; i <= n; i++) {
 System.out.println("Enter element " + i + ":");
 a[i] = scanner.nextInt();
 }
 // Build the max heap
 buildMaxHeap(a, n);
 // Output the max heap
 System.out.println("Max Heap:");
 for (int i = 1; i <= n; i++) {
 System.out.println(a[i]);
 }
 scanner.close();
 }
}
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

Output :

```
Roll no: 15
Enter number of elements in the array:
4
Enter element 1:
12
Enter element 2:
23
Enter element 3:
34
Enter element 4:
45
Max Heap:
45
23
34
12
```

PRACTICAL NO:- 09

Aim:-Implementation of Adjacency matrix.

Source Code:

```
import java.util.Scanner;
class AdjacencyMatrix {
 private int[][] adj;
 private int n;
 // Constructor to initialize the adjacency matrix
 public AdjacencyMatrix(int n) {
 this.n = n;
 adj = new int[n][n];
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
// Initialize all values to 0
for (int i = 0; i < n; i++) {
for (int j = 0; j < n; j++) {
adj[i][j] = 0;
} } }
// Method to add an edge
public void addEdge(int origin, int dest) {
if (origin >= n || dest >= n || origin < 0 || dest < 0) {
System.out.println("Wrong nodes");
} else {
adj[origin][dest] = 1;
} }
// Method to display the adjacency matrix
public void display() {
for (int i = 0; i < n; i++) {
for (int j = 0; j < n; j++) {
System.out.print(adj[i][j] + "\t");
}
System.out.println();
} }
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
int nodes, maxEdges, origin, dest;
System.out.println("Roll no. 15");
System.out.print("Enter Maximum number of nodes: ");
nodes = scanner.nextInt();
AdjacencyMatrix am = new AdjacencyMatrix(nodes);
maxEdges = nodes * (nodes - 1);
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
System.out.println("Enter -1 -1 to exit");
for (int i = 0; i < maxEdges; i++) {
System.out.print("\nEnter edges: ");
origin = scanner.nextInt();
dest = scanner.nextInt();
if (origin == -1 && dest == -1) {
break;
} else {
am.addEdge(origin, dest);
} }
am.display();
scanner.close();
} }
```

Output :

```
Roll no. 15
Enter Maximum number of nodes: 4
Enter -1 -1 to exit

Enter edges: 0 1

Enter edges: 1 0

Enter edges: 1 1

Enter edges: 0 0

Enter edges: -1 -1
1       1       0       0
1       1       0       0
0       0       0       0
0       0       0       0
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

PRACTICAL NO:- 10

Aim:-Implementation of Graph traversal. (DFS and BFS).

i)BFS

Source Code:

```
import java.util.*;
public class GraphBFS {
   static final int NODE = 6; // Number of nodes in the graph
   static int[][] graph = {
      {0, 1, 1, 1, 0, 0},
      {1, 0, 0, 1, 1, 0},
      {1, 0, 0, 1, 0, 1},
      {1, 1, 1, 0, 1, 1},
      {0, 1, 0, 1, 0, 1},
      {0, 0, 1, 1, 1, 0}
   };
   // Node class to represent a vertex
   static class Node {
      int val;     // Vertex index
      boolean visited; // Visited state
      public Node(int val) {
         this.val = val;
         this.visited = false;
      }    }
   // BFS function
   public static void bfs(Node[] vertices, Node start) {
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
        Queue<Node> queue = new LinkedList<>();
        // Mark the start node as visited and enqueue it
        start.visited = true;
        queue.add(start);
        System.out.print("BFS Traversal: ");
        // Perform BFS
        while (!queue.isEmpty()) {
            Node current = queue.poll(); // Dequeue the node
            System.out.print((char) (current.val + 'A') + " "); // Print the node
            // Visit all neighbors
            for (int i = 0; i < NODE; i++) {
                if (graph[current.val][i] == 1 && !vertices[i].visited) { // Edge exists and not visited
                    vertices[i].visited = true; // Mark as visited
                    queue.add(vertices[i]); // Enqueue the neighbor
                }   }   }
        System.out.println(); // New line after traversal
    }
    public static void main(String[] args) {
        System.out.println("Roll no. 60");
        System.out.println("Graph Traversal using BFS.\n");
        Node[] vertices = new Node[NODE];
        // Initialize all vertices
        for (int i = 0; i < NODE; i++) {
            vertices[i] = new Node(i);
        }
        // Starting node: B (index 1)
        char startChar = 'B';
        Node startNode = vertices[startChar - 'A'];
```

```
        // Perform BFS starting from the given node
        bfs(vertices, startNode);
    }   }
```

Output :

```
Roll no. 15
Graph Traversal using BFS.

BFS Traversal: B A D E C F
```

ii) DFS

Source Code:

```
import java.util.*;
class Graph {
 // Adjacency list representation of the graph
 private final Map<Integer, List<Integer>> adj;
 // Constructor to initialize the graph
 public Graph() {
 adj = new HashMap<>();
 }
 // Function to add an edge to the graph
 public void addEdge(int v, int w) {
 adj.putIfAbsent(v, new ArrayList<>());
 adj.get(v).add(w);
 }
 // Recursive DFS function
 public void DFS(int v, Set<Integer> visited) {
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
// Mark the current node as visited and print it
visited.add(v);
System.out.print(v + " ");
// Recur for all the vertices adjacent to this vertex
for (int neighbor : adj.getOrDefault(v, new ArrayList<>())) {
if (!visited.contains(neighbor)) {
DFS(neighbor, visited);
} } }
public static void main(String[] args) {
   System.out.println("Roll no. 15");
// Create a new graph
Graph g = new Graph();
// Add edges to the graph
g.addEdge(0, 1);
g.addEdge(0, 2);
g.addEdge(1, 2);
g.addEdge(2, 0);
g.addEdge(2, 3);
g.addEdge(3, 3);
// Create a set to track visited nodes
Set<Integer> visited = new HashSet<>();
// Print DFS traversal starting from vertex 2
System.out.println("Following is Depth First Traversal (starting from vertex 2):");
g.DFS(2, visited);
} }
```

Output :

```
Roll no. 15
Following is Depth First Traversal (starting from vertex 2):
2 0 1 3
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

PRACTICAL NO:- 11

Aim:-Create a minimum spanning tree using any method Kruskal's Algorithm or Prim's Algorithm.

Source Code:

```
import java.util.*;
public class KruskalsMST {
 static int[] father; // Array to store the parent of each node
 static edge[] tree; // Array to store the edges in the MST
 static int wt_tree = 0; // To store the weight of the spanning tree
 static int cnt = 0; // To count the number of edges in the spanning tree
 static int n; // Number of nodes
 // Edge class to store edges
 static class edge {
 int u, v, weight;
 }
 // Method to find the root of a node using path compression
 public static int find(int i) {
 if (father[i] < 0) {
 return i; // If parent is negative, i is the root
 }
 father[i] = find(father[i]); // Path compression
 return father[i];
 }
 // Method to union two sets by rank
 public static void union(int root1, int root2) {
 if (father[root1] < father[root2]) { // Union by rank
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
father[root1] += father[root2];
father[root2] = root1;
} else {
father[root2] += father[root1];
father[root1] = root2;
}
}
// Method to insert edges into the priority queue
public static void insert_pque(int i, int j, int wt, PriorityQueue<edge> pq) {
edge tmp = new edge();
tmp.u = i;
tmp.v = j;
tmp.weight = wt;
pq.add(tmp); // Insert edge into the priority queue
}
// Method to process the priority queue and form the MST
public static void make_tree(PriorityQueue<edge> pq) {
edge tmp;
int node1, node2, root_n1, root_n2;
// While there are edges and we haven't added n-1 edges to the tree
while (cnt < n - 1 && !pq.isEmpty()) {
tmp = pq.poll(); // Get the minimum edge
node1 = tmp.u;
node2 = tmp.v;
root_n1 = find(node1); // Find the roots of the two nodes
root_n2 = find(node2);
// If they belong to different sets, add the edge to the MST
if (root_n1 != root_n2) {
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
insert_tree(tmp.u, tmp.v, tmp.weight); // Insert the edge into the MST
wt_tree += tmp.weight; // Add the weight of the edge
union(root_n1, root_n2); // Union the sets
}
}
}
// Method to insert an edge into the MST
public static void insert_tree(int i, int j, int wt) {
cnt++; // Increment the edge count
tree[cnt] = new edge();
tree[cnt].u = i;
tree[cnt].v = j;
tree[cnt].weight = wt;
}
// Method to create the graph, input edges, and set up the priority queue
public static void create_graph(Scanner sc, PriorityQueue<edge> pq) {
System.out.print("Enter the number of nodes: ");
n = sc.nextInt(); // Read the number of nodes
father = new int[n + 1]; // Initialize the father array for Union-Find
Arrays.fill(father, -1); // Initially, every node is its own parent
tree = new edge[n]; // Initialize the tree array to store MST edges
System.out.println("Enter edges (0 0 to quit) weight: ");
int origin, destin, wt;
while (true) {
System.out.print("Enter origin and destination (0 0 to quit): ");
origin = sc.nextInt();
destin = sc.nextInt();
if (origin == 0 && destin == 0) break; // End input if origin and destination are 0
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
System.out.print("Enter weight for this edge: ");
wt = sc.nextInt();
if (origin > n || destin > n || origin <= 0 || destin <= 0) {
System.out.println("Invalid edge");
continue;
}
insert_pque(origin, destin, wt, pq); // Insert the edge into the priority queue
}
if (pq.size() < n - 1) {
System.out.println("Spanning tree is not possible.");
System.exit(1);
}
}
// Method to display the edges in the MST and the total weight
public static void display_mst() {
System.out.println("\nEdges to be included in spanning tree:");
for (int i = 1; i <= cnt; i++) {
System.out.println(tree[i].u + " - " + tree[i].v + " (Weight: " + tree[i].weight + ")");
}
System.out.println("Weight of this spanning tree is: " + wt_tree);
}
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
PriorityQueue<edge> pq = new PriorityQueue<>(Comparator.comparingInt(e -> e.weight)); //
Priority queue to store edges sorted by weight
System.out.println("Rollno: 15");
// Create graph and input edges
create_graph(sc, pq);
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

```
// Create the MST
make_tree(pq);
// Display the result
display_mst();
sc.close();
}
}
```

Output :

```
Roll No: 15

Enter the number of nodes: 4
Enter edges (format: origin destination weight, 0 0 0 to quit):
1 2 1
1 3 3
2 3 2
3 4 4
0 0 0

Kruskal's Minimum Spanning Tree Algorithm

Edges in the Minimum Spanning Tree:
1 - 2 (Weight: 1)
2 - 3 (Weight: 2)
3 - 4 (Weight: 4)
Total weight of the Minimum Spanning Tree: 7
```

Name: Vishwajit Gadale
Roll No: 15
FYMCA

PRACTICAL NO:- 12

Aim:-Group Name and Project Name.

PROJECT NAME : Quiz Application.

GROUP NAME : B 06 Pratik Bhalerao

B 15 Vishwajit Gadale

B 66 Akash Vishwakarma

B 74 Amar Bharati