National University of Computer & Emerging Sciences,
Karachi
Computer Science Department
Fall 2022, Lab Manual – 08

Course Code: CL-2001	Course : Data Structures - Lab
Instructor(s) :	Abeer Gauher, Sobia Iftikhar

LAB - 8

Implementation of Stacks, Queue, binary tree nodes and nodes for general tree, Traverse the tree with the three common orders

Objective: {Stack with Array and Linked list , Application of Stack, Queue with Array and Linked List , Application of Queue, implement classes for binary tree nodes and nodes for general tree, Traverse the tree with the three common orders }

Lab Tasks:

Stack with Array

```
class Stacks {
private int arr[];
private int top;
private int capacity;

// Creating a stack
Stacks(int size) {
 arr = new int[size];
 capacity = size;
 top = -1;
}

// Add elements into stack
public void push(int x) {
 if (isFull()) {
System.out.println("OverFlow\\n");
   System.exit(1);
 }

 System.out.println("Inserting " + x);
 arr[++top] = x;
}

// Remove element from stack
public int pop() {
 if (isEmpty()) {
   System.out.println("STACK EMPTY");
   System.exit(1);
 }
 return arr[top--];
}

// Utility function to return the size of the stack


// Check if the stack is empty
public Boolean isEmpty() {
 return top == -1;
}

// Check if the stack is full
public Boolean isFull() {
 return top == capacity - 1;
}

public void printStack() {
 for (int i = 0; i <= top; i++) {
   System.out.println(arr[i]);
 }
}

public static void main(String[] args) {
 Stacks stack = new Stacks(5);

 stack.push(1);
 stack.push(2);
 stack.push(3);
 stack.push(4);

 stack.pop();
 stack.printStack();
 System.out.println("\nAfter popping out");

 while (!stack.isEmpty()) {
    System.out.printf(" %d", stack.pop());
 }

 stack.printStack();

}
}
```

Task-1: use above code as sample

Consider a empty stack of integers. Let the numbers 1,2,3,4,5,6 be pushed on to this stack only in the order they appeared from left to right. Let S indicates a push and X indicate a pop operation. Can they be permuted in to the order 325641(output) and order 154623(output)? (Hint: SSSSSSXXXXXX outputs 654321)

Stack with Linked list

```
        // Driver code
class xyz {
public static void main(String[] args){
StackUsingLinkedlist obj = new
        StackUsingLinkedlist();
                obj.push(11);
                obj.push(22);
                obj.display();
System.out.printf("\nTop element is %d\n",
        obj.peek());
        obj.pop();}}
class StackUsingLinkedlist { // A linked list node
        private class Node {
        int data;  Node link; }
        Node top;
        StackUsingLinkedlist() { this.top = null;
        }
        public void push(int x) {
                Node temp = new Node();
        if (temp == null) {
        System.out.print("\nHeap Overflow");
                return; }
                temp.data = x;
                temp.link = top;
                top = temp; }
        public boolean isEmpty() {
              return top == null; }
        public int peek()
        {       if (!isEmpty()) {
                return top.data; }
                else {
        System.out.println("Stack is empty");
              return -1;} }
        public void pop(){
              if (top == null) {
        System.out.print("\nStack Underflow");
                      return; }
                top = (top).link; }
        public void display()
        { if (top == null) {
        System.out.printf("\nStack Underflow");
                exit(1); }
        else {
                Node temp = top;
                while (temp != null) {
                System.out.print(temp.data);
                      temp = temp.link;
                      if(temp != null)
        System.out.print(" -> ");
                      }
                }
        }
}
```

Task-2:

A. Design a Main class which creates a ShoppingCart with limit of 9 items, fill the ShoppingCart with 9 items after that the user performs the below task:

1. Insert 10th item in the stack.
2. Remove the Inserted values till the Last value and print the message that the stack is empty.

Application of Stack (convert infix expression to postfix)

Sample Pseudocode

```
Begin
  initially push some special character say # into the stack
  for each character ch from infix expression, do
    if ch is alphanumeric character, then
      add ch to postfix expression
    else if ch = opening parenthesis (, then
      push ( into stack
    else if ch = ^, then          //exponential operator of higher precedence
      push ^ into the stack
    else if ch = closing parenthesis ), then
        while stack is not empty and stack top ≠ (,
        do pop and add item from stack to postfix expression
      done
      pop ( also from the stack
    else
      while stack is not empty AND precedence of ch <= precedence of stack top element, do
        pop and add into postfix expression
      done
      push the newly coming character.
  done
  while the stack contains some remaining characters, do
    pop and add to the postfix expression
  done
  return postfix
End
```

Code Snippet

```
//Function to return precedence of operators
int prec(char c)
{
if(c == '^')
return 3;
else if(c == '*' || c == '/')
return 2;
else if(c == '+' || c == '-')
return 1;
else
return -1;
}
main()
{
   string exp = "a+b*(c^d-e)^(f+g*h)-i";
   infixToPostfix(exp);
   return 0;
```

Task-3:

Use the Upper code snippet implement the utility function with the help of array based stack **infixToPostfix** by using sample pseudocode

Queue with Array

```
class Queue {
        static private int front, rear, capacity;
        static private int queue[];
        Queue(int c)
        {
                front = rear = 0;
                capacity = c;
                queue = new int[capacity];
        }
}
public class StaticQueueinjava {

        // Driver code
        public static void main(String[] args)
        {
                Queue q = new Queue(5);
                // print Queue elements
                q.queueDisplay();
                q.queueEnqueue(20);
                q.queueEnqueue(30);
                q.queueEnqueue(40);
                q.queueEnqueue(50);
                q.queueDisplay();
                q.queueEnqueue(60);

                q.queueDisplay();

                q.queueDequeue();
                q.queueDequeue();
                System.out.printf( "\n\nafter two
node deletion\n\n");
                q.queueDisplay();
                q.queueFront();
        }
```

Task-4:

A. Use the Upper code snippet implement the following utility function in the Array based Queue
   1. Write a implementation of **queueFront**
   2. Write a implementation of **queueDisplay**
   3. Write a function **queueEnqueue** when a new integer value is added in the array
   4. Write a function **queueDequeue** when any data member is removed from the queue, space complexity should not be compromised.
   5. Display the output in the manner of Function calls.

Circular Queue

Task-5:

1. Write a function for circular queue
2. call enqueue function three times for ‘A’, ‘B’, ‘C’
3. Dequeue the ‘A’
4. Enqueue the ‘D’
5. Print the Dequeue that must B
6. Enqueue the ‘E’
7. Print the Dequeue that must be C

Binary Tree

```
class Node {
int key;
Node left, right;

public Node(int item)
{
key = item;
left = right = null;
}
}

// A Java program to introduce Binary Tree
class BinaryTree {

Node root;

// Constructors
BinaryTree(int key) { root = new Node(key); }
BinaryTree() { root = null; }
// Method 02
                public Nod create() {
                        Scanner sc = new Scanner(System.in);
                        Nod root = null;
                        System.out.println("Enter data: ");
                        int data = sc.nextInt();
                        if(data == -1) return null;
                        root = new Nod(data);
                        System.out.println("Enter left for " + data);
                        root.left = create();
                        System.out.println("Enter right for "+ data);
                        root.right = create();
                        return root; }
                public void traversePreOrder(Nod node) {

                if (node != null) {
                 System.out.print(" " + node.key);
                 traversePreOrder(node.left);
                 traversePreOrder(node.right);
                }
                }
public static void main(String[] args)
{
BinaryTree tree = new BinaryTree();
//Method 01
                tree.root = new Node(1);
tree.root.left = new Node(2);
```

```
tree.root.right = new Node(3);
tree.root.left.left = new Node(4);

}}
```

Task-6:

1. Write a function to construct a binary tree
2. Data must be provided by user at the run time
3. -1 input from user indicate the is no chil for particular node
4. Print the Binary tree in format of “Preorder, Inorder”