**ADS Module Exam PRN - 220940320077**

- - - - - - - - - **- - - - - - - - - - - - - - - - - - - - -- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -**

**Question 1) Given a string S, remove consecutive duplicates from it recursively.**

Solution:

import java.util.\*;

class StringDuplicate

{

static String removeDuplicate(String str) //abcc

{

if(str.length() == 1)

return str;

else if(str.charAt(0) == str.charAt(1))

return removeDuplicate(str.substring(1));

else

return str.charAt(0)+removeDuplicate(str.substring(1));

}

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

System.out.println("Enter a String :");

String str = sc.next();

String out = removeDuplicate(str);

System.out.println("Output String :");

System.out.println(out);

}

}

Output:

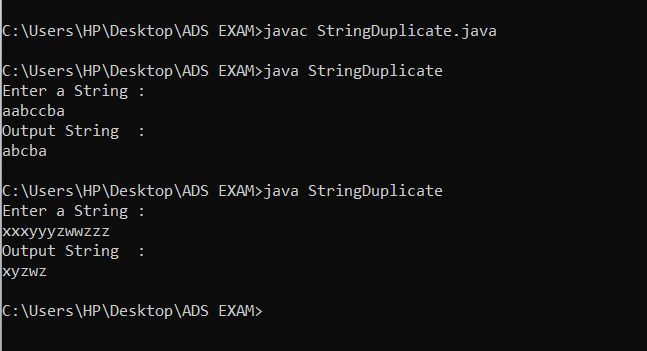
Enter a String :

aabccba

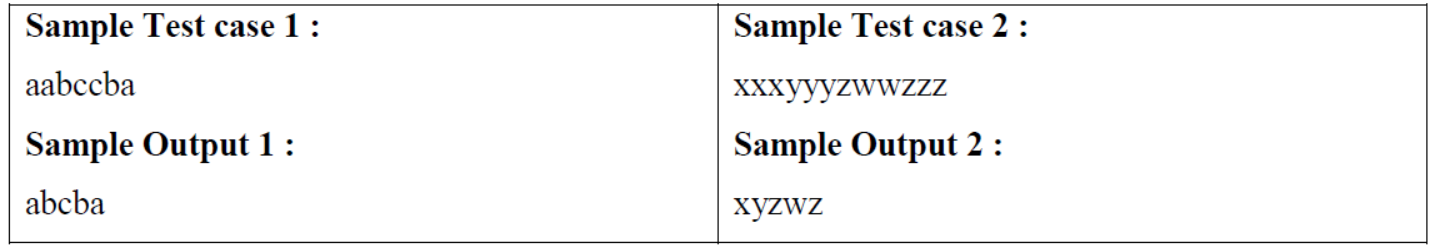
Output String :

abcba

CMD Snapshot:



Test Cases:



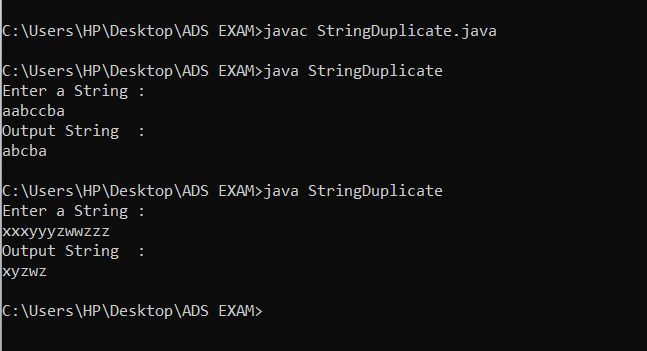
Test Case 1 ):

Enter a String :

aabccba

Output String :

abcba



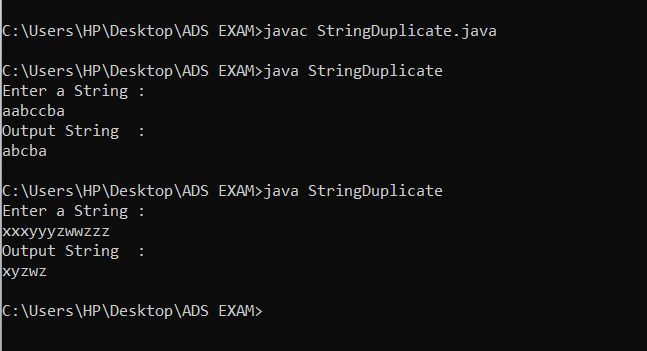
Test Case 2 ):

Enter a String :

xxxyyyzwwzzz

Output String :

xyzwz



**Question 2): For a given string expression containing only round brackets or parentheses, check if they are balanced or not. Brackets are said to be balanced if the bracket which opens last, closes first.**

**Example: Expression: (()())**

**Since all the opening brackets have their corresponding closing brackets, we say it is balanced and hence the output will be, 'true'.You need to return a boolean value indicating whether the expression is balanced or not.**

Solution:

import java.util.\*;

class Stack

{

int size=100;

int top=-1;

char[] arr = new char[size];

Scanner sc = new Scanner(System.in);

void push(char c)

{

top=top+1;

arr[top]=c;

}

void pop(char c)

{

top=top-1;

}

boolean isEmpty()

{

if(top==-1)

return true;

return false;

}

boolean isBalanced(String str)

{

char[] arr = str.toCharArray();

for(char c : arr)

{

if(c == '(')

{

push(c);

}

else

{

if(isEmpty())

{

return false;

}

else

{

char t = arr[top];

if (c == ')')

pop(c);

else

return false;

}

}

}

return isEmpty();

}

}

class BalancedParentheses

{

public static void main(String args[])

{

Stack stack = new Stack();

Scanner sc = new Scanner(System.in);

System.out.println("Enter a String expression :");

String str = sc.next();

System.out.println(stack.isBalanced(str));

}

}

Output:

C:\Users\HP\Desktop\ADS EXAM>java BalancedParentheses

Enter a String expression :

(()()())

true

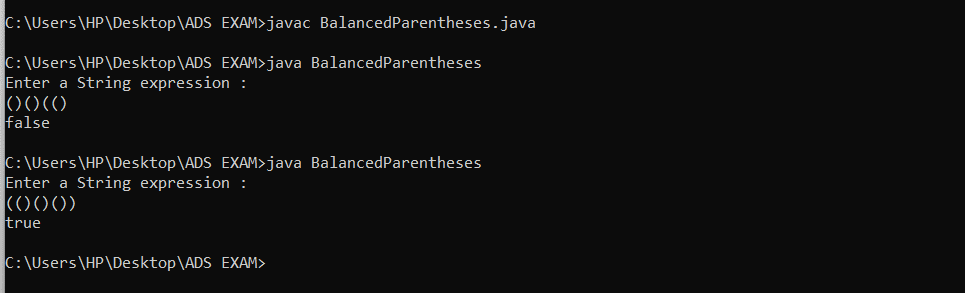
C:\Users\HP\Desktop\ADS EXAM>java BalancedParentheses

Enter a String expression :

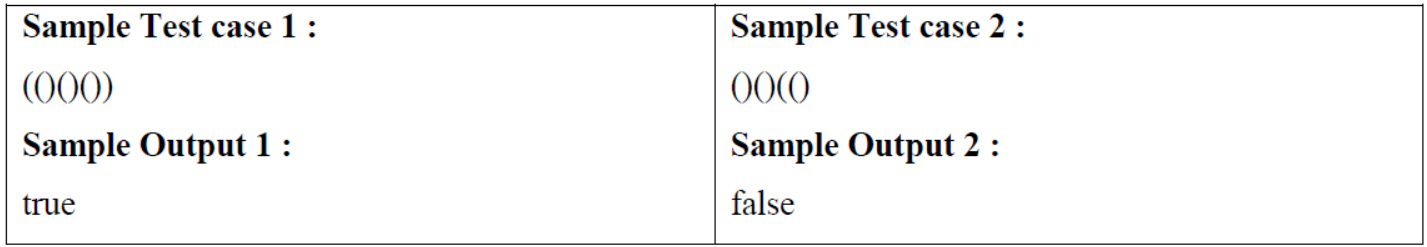
()()(()

false

CMD Snapshot:



Test Cases:



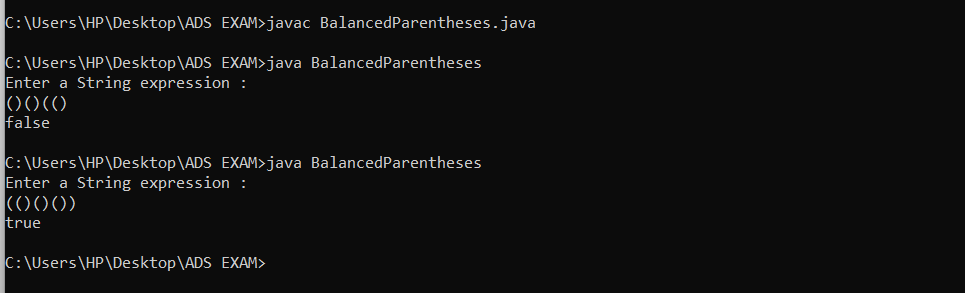
Test Case 1 ):

C:\Users\HP\Desktop\ADS EXAM>java BalancedParentheses

Enter a String expression :

(()()())

true



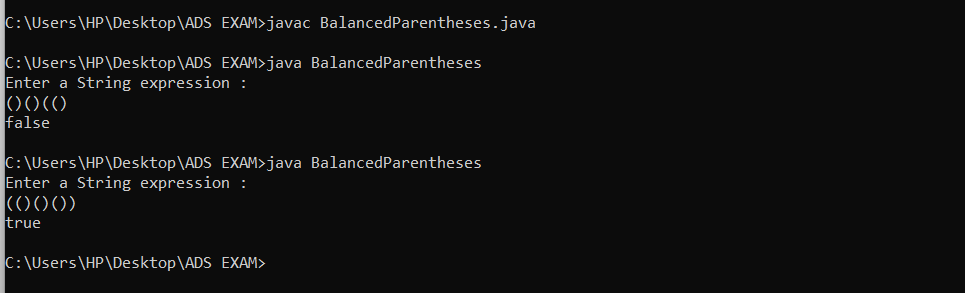
Test Case 2 ):

C:\Users\HP\Desktop\ADS EXAM>java BalancedParentheses

Enter a String expression :

()()(()

false



**Q3. Write a menu driven java program to design the linked list functionality as follows:**

**1. Create a linked list.**

**2. Reverse the linked list.**

**3. Eliminate duplicates from the list.**

**4. Check whether the given list is a palindrome or not.**

**Solution :**

import java.util.\*;

class LinkedListDemo

{

Node head = null;

Node slow\_ptr, fast\_ptr, second\_half;

static class Node

{

int data;

Node next;

Node()

{

Node next=null;

}

Node(int data)

{

this.data=data;

this.next=null;

}

}

void create(Scanner sc)

{

int data,m;

do

{

System.out.println("Enter data : ");

data=sc.nextInt();

Node new\_node = new Node(data);

if(head == null)

{

head=new\_node;

}

else

{

Node temp = head;

while(temp.next!=null)

{

temp=temp.next;

}

temp.next=new\_node;

}

System.out.println("Enter 0 to add more data :");

m = sc.nextInt();

}while(m==0);

}

Node reverse()

{

Node prev = null;

Node next = null;

Node current = head;

while(current != null)

{

next=current.next;

current.next=prev;

prev = current;

current= next;

}

head=prev;

return head;

}

void eliminateDuplicates()

{

Node temp1=head;

Node temp2 = null;

while(temp1 != null && temp1.next != null)

{

temp2=temp1;

while(temp2.next != null)

{

if(temp1.data==temp2.next.data)

{

temp2.next= temp2.next.next;

}

else

{

temp2= temp2.next;

}

}

temp2 = temp2.next;

}

}

boolean isPalindrome()

{

slow\_ptr= head;

fast\_ptr= head;

Node prev\_slow = head;

Node mid = null;

boolean result = true;

if(head != null && head.next!=null)

{

while(fast\_ptr != null && fast\_ptr.next != null)

{

fast\_ptr = fast\_ptr.next.next;

prev\_slow = slow\_ptr;

slow\_ptr = slow\_ptr.next;

}

if(fast\_ptr != null)

{

mid = slow\_ptr;

slow\_ptr=slow\_ptr.next;

}

second\_half = slow\_ptr;

prev\_slow.next = null;

reverse();

result= compare(head,second\_half);

reverse();

if(mid !=null)

{

prev\_slow.next=mid;

mid.next=second\_half;

}

else

prev\_slow.next = second\_half;

}

return result;

}

boolean compare(Node head1, Node head2)

{

Node temp\_1 = head1;

Node temp\_2 = head2;

while(temp\_1 != null && temp\_2 != null)

{

if(temp\_1.data == temp\_2.data)

{

temp\_1= temp\_1.next;

temp\_2= temp\_2.next;

}

else

return false;

}

if(temp\_1 == null && temp\_2 == null)

return true;

return false;

}

void print()

{

Node temp=head;

if(head == null)

{

System.out.println("LinkedList doesnt exits ");

}

else

{

while(temp != null)

{

System.out.print(temp.data+" ->");

temp=temp.next;

}

System.out.print(" Null");

System.out.println();

}

}

public static void main(String args[])

{

LinkedListDemo list = new LinkedListDemo();

Scanner sc = new Scanner(System.in);

int m;

do

{

System.out.println("Enter choice for Operation : ");

System.out.println("1.Create 2.Reverse 3.Eliminate Duplicates 4.Check Palindrome ");

int choice = sc.nextInt();

switch(choice)

{

case 1:

list.create(sc);

System.out.println("Linked list :");

list.print();

break;

case 2:

list.reverse();

System.out.println("Reversed Linked list :");

list.print();

list.reverse();

break;

case 3:

list.eliminateDuplicates();

list.print();

break;

case 4:

System.out.println("Is Linked list Palindrome :" +list.isPalindrome());

break;

default:

break;

}

System.out.println("Enter 0 to go back to the Menu :");

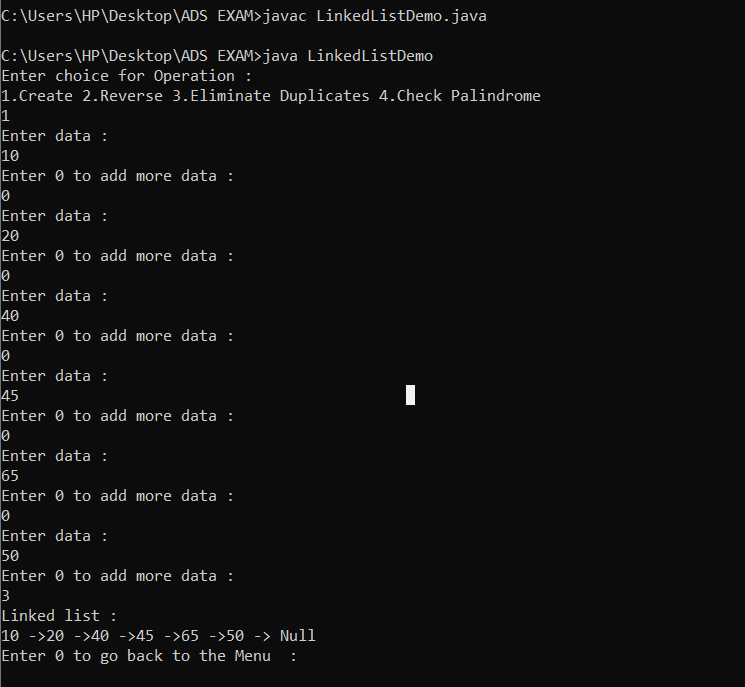
m = sc.nextInt();

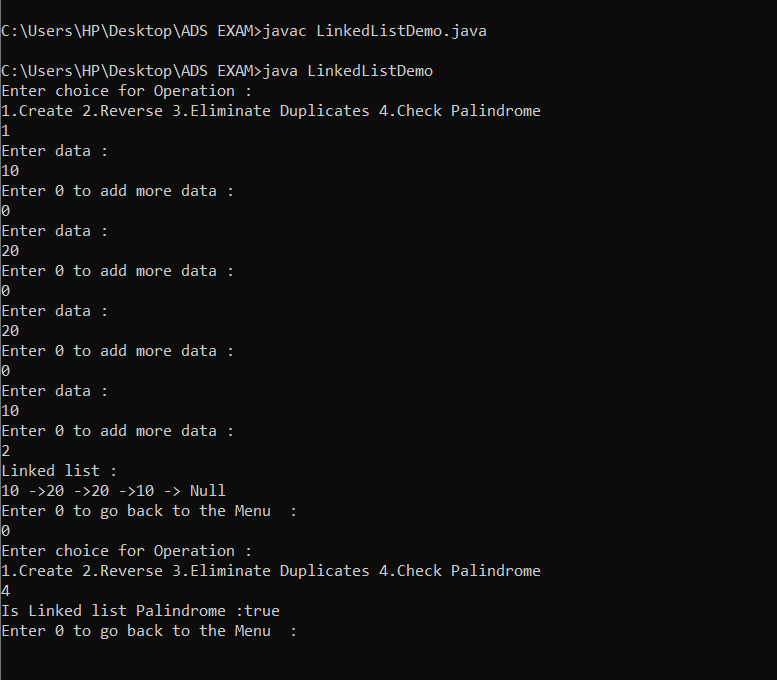
}while(m==0);

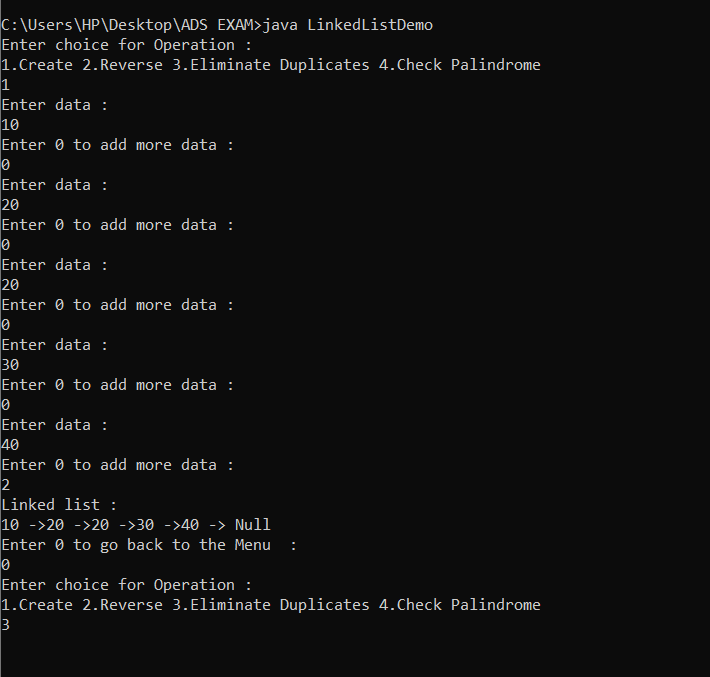
}

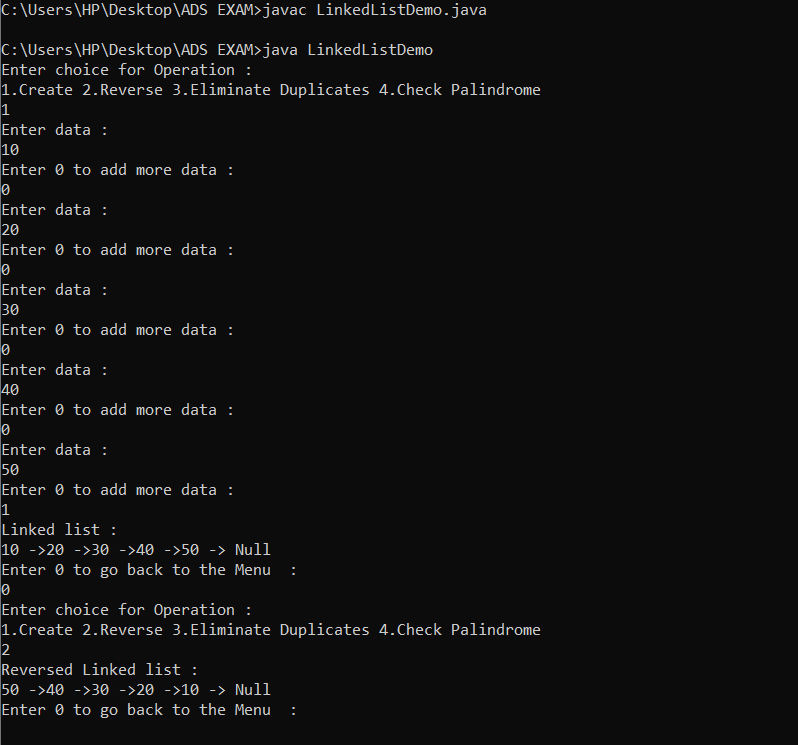
}

**1. Create**

****

2. Palindrome

3. Duplicates

Reverse