Swapnanil Dutta Roll: 12

1.

Code:

```
import java.util.Iterator;
import java.util.LinkedList;

public class LinkedListEx {
    public static void main(String[] args) {
        LinkedList<Integer> list = new LinkedList<Integer>();
        int size;
        Iterator iterator;
        list.add(11);
        list.add(22);
        list.add(33);
        list.add(44);
        size = list.size();
        System.out.print("Linked list data: ");
        iterator = list.iterator();
        while (iterator.hasNext()) {
            System.out.print(iterator.next() + " ");
        }
        System.out.println();
        if (list.isEmpty()) {
            System.out.println("Linked list is empty");
        } else {
            System.out.println("Linked list size: " + size);
        }
        list.addFirst(55);
        System.out.println("Adding data at first location: 55");
        System.out.print("Now the list contains: ");
        iterator = list.iterator();
        while (iterator.hasNext()) {
            System.out.print(iterator.next() + " ");
        }
        System.out.println();
        System.out.println("Now the size of list: " + list.size());
        list.addLast(66);
        System.out.println("Adding data at last location: 66");
        System.out.print("Now the list contain: ");
        iterator = list.iterator();
```

```
        while (iterator.hasNext()) {
            System.out.print(iterator.next() + " ");
        }
        System.out.println();
        System.out.println("Now the size of list: " + list.size());
        list.add(2, 99);
        System.out.println("Adding data at 3rd location: 99");
        System.out.print("Now the list contains: ");
        iterator = list.iterator();
        while (iterator.hasNext()) {
            System.out.print(iterator.next() + " ");
        }
        System.out.println();
        System.out.println("Now the size of list: " + list.size());
        System.out.println("First data: " + list.getFirst());
        System.out.println("Last data: " + list.getLast());
        System.out.println("Data at position 3 is: " + list.get(3));
        int first = list.removeFirst();
        System.out.println("Data removed from 1st location: " + first);
        System.out.print("Now the list contains: ");
        iterator = list.iterator();
        while (iterator.hasNext()) {
            System.out.print(iterator.next() + " ");
        }
        System.out.println();
        System.out.println("Now the size of list: " + list.size());
        int last = list.removeLast();
        System.out.println("Data removed from last location: " + last);
        System.out.print("Now the list contains: ");
        iterator = list.iterator();
        while (iterator.hasNext()) {
            System.out.print(iterator.next() + " ");
        }
        System.out.println();
        System.out.println("Now the size of list: " + list.size());
        int second = list.remove(1);
        System.out.println("Data removed from 2nd location: " + second);
        System.out.print("Now the list contains: ");
        iterator = list.iterator();
        while (iterator.hasNext()) {
```

```
            System.out.print(iterator.next() + " ");
        }
        System.out.println();
        System.out.println("Now the size of list: " + list.size());
        list.clear();
        if (list.isEmpty()) {
            System.out.println("Linked list is empty");
        } else {
            System.out.println("Linked list size: " + size);
        }
    }
}
```

Output:

```
PS D:\OOPS-PCC-CS593\Day-31-(17.02.2021)> javac LinkedListEx.java
PS D:\OOPS-PCC-CS593\Day-31-(17.02.2021)> java LinkedListEx
Linked list data: 11 22 33 44
Linked list size: 4
Adding data at first location: 55
Now the list contains: 55 11 22 33 44
Now the size of list: 5
Adding data at last location: 66
Now the list contain: 55 11 22 33 44 66
Now the size of list: 6
Adding data at 3rd location: 99
Now the list contains: 55 11 99 22 33 44 66
Now the size of list: 7
First data: 55
Last data: 66
Data at position 3 is: 22
Data removed from 1st location: 55
Now the list contains: 11 99 22 33 44 66
Now the size of list: 6
Data removed from last location: 66
Now the list contains: 11 99 22 33 44
Now the size of list: 5
Data removed from 2nd location: 99
Now the list contains: 11 22 33 44
Now the size of list: 4
Linked list is empty
```

2.
Code:

```
import java.util.Iterator;
import java.util.Stack;

public class StackEx {
    public static void main(String[] args) {
        Stack stack = new Stack<>();
        Iterator iterator;
        custom_push(stack);
        iterator = stack.iterator();
        System.out.println("\nStack elements: ");
        while (iterator.hasNext()) {
            System.out.print(iterator.next() + " ");
        }
        System.out.println();
        custom_pop(stack);
        System.out.println(custom_empty(stack));
        custom_push(stack);
        iterator = stack.iterator();
        System.out.println("\nStack elements: ");
        while (iterator.hasNext()) {
            System.out.print(iterator.next() + " ");
        }
        System.out.println();
        custom_peek(stack);
        custom_search(stack, 4);
        custom_search(stack, 7);
        System.out.println(custom_empty(stack));
    }

    static void custom_push(Stack st) {
        System.out.println("\nPush operation:~ ");
        for (int i = 1; i <= 5; i++) {
            System.out.println("Item pushed: " + i);
            st.push(i);
        }
    }

    static void custom_pop(Stack st) {
```

```
        System.out.println("\nPop operation:~ ");
        for (int i = 1; i <= 5; i++) {
            Integer item = (Integer) st.pop();
            System.out.println("Item popped: " + item);
        }
    }

    static void custom_peek(Stack st) {
        System.out.println("\nPeek operation:~ ");
        Integer element = (Integer) st.peek();
        System.out.println("Element on stack top : " + element);
    }

    static void custom_search(Stack st, int element) {
        System.out.println("\nSearch operation:~ ");
        Integer pos = (Integer) st.search(element);
        if (pos == -1) {
            System.out.println("Element " + element + " is not found");
        } else {
            System.out.println("Element " + element + " is found at
position " + pos);
        }
    }

    static String custom_empty(Stack st) {
        System.out.println("\nChecking stack is empty or not:~ ");
        if (st.empty() == true)
            return "Stack is empty!";
        else
            return "Stack is not empty!";
    }
}
```

Output:

```
PS D:\OOPS-PCC-CS593\Day-31-(17.02.2021)> javac StackEx.java
Note: StackEx.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
PS D:\OOPS-PCC-CS593\Day-31-(17.02.2021)> java StackEx

Push operation:~
Item pushed: 1
Item pushed: 2
Item pushed: 3
Item pushed: 4
Item pushed: 5

Stack elements:
1 2 3 4 5

Pop operation:~
Item popped: 5
Item popped: 4
Item popped: 3
Item popped: 2
Item popped: 1

Checking stack is empty or not:~
Stack is empty!

Push operation:~
Item pushed: 1
Item pushed: 2
Item pushed: 3
Item pushed: 4
Item pushed: 5

Stack elements:
1 2 3 4 5

Peek operation:~
Element on stack top : 5

Search operation:~
Element 4 is found at position 2

Search operation:~
Element 7 is not found

Checking stack is empty or not:~
Stack is not empty!
```

3.

Code:

```java
import java.util.*;

public class QueueEx {
    public static void main(String[] args) throws InterruptedException {
        Queue<Integer> q = new ArrayDeque<Integer>();
        for (Integer i = 1; i <= 5; i++) {
            q.add(i);
        }
        System.out.println("The elements of the queue- " + q);
        int rem = q.remove();
        System.out.println("The removed Element- " + rem);
        System.out.println("The elements of the queue- " + q);
        System.out.println("The head of the queue- " + q.peek());
        System.out.println("The size of the Queue-" + q.size());
    }
}
```

Output:

```
PS D:\OOPS-PCC-CS593\Day-31-(17.02.2021)> javac QueueEx.java
PS D:\OOPS-PCC-CS593\Day-31-(17.02.2021)> java QueueEx
The elements of the queue- [1, 2, 3, 4, 5]
The removed Element- 1
The elements of the queue- [2, 3, 4, 5]
The head of the queue- 2
The size of the Queue-4
```

4.

Code:

```
public class EnumEx {
    public static void main(String[] args) {
        System.out.println("All months in a year:~ ");
        for (Months m : Months.values()) {
            System.out.println(m + " is number " + m.getNumber() + " month
and has index-" + m.ordinal() + ".");
        }
        Months m1 = Months.valueOf("OCTOBER");
        System.out.print(m1);
    }
}

enum Months {
    JANUARY(1), FEBRUARY(2), MARCH(3), APRIL(4), MAY(5), JUNE(6), JULY(7),
AUGUST(8), SEPTEMBER(9), OCTOBER(10),
    NOVEMBER(11), DECEMBER(12);

    private int number;

    Months(int n) {
        number = n;
    }

    public int getNumber() {
        return number;
    }
}
```

Output:

```
PS D:\OOPS-PCC-CS593\Day-31-(17.02.2021)> javac EnumEx.java
PS D:\OOPS-PCC-CS593\Day-31-(17.02.2021)> java EnumEx
All months in a year:~
JANUARY is number 1 month and has index-0.
FEBRUARY is number 2 month and has index-1.
MARCH is number 3 month and has index-2.
APRIL is number 4 month and has index-3.
MAY is number 5 month and has index-4.
JUNE is number 6 month and has index-5.
JULY is number 7 month and has index-6.
AUGUST is number 8 month and has index-7.
SEPTEMBER is number 9 month and has index-8.
OCTOBER is number 10 month and has index-9.
NOVEMBER is number 11 month and has index-10.
DECEMBER is number 12 month and has index-11.
OCTOBER
```

5.
Code:

```
public class ObjectCloningEx {
    public static void main(String[] args) {
        Employee e1 = new Employee(101, "Rohit", "Delhi", 95725.75);
        System.out.println(e1);
        Employee e2 = null;
        try {
            e2 = (Employee) e1.clone();
        } catch (CloneNotSupportedException ex) {
            System.out.println(ex);
        }
        if (e1 == e2)
            System.out.println("same memory location");
        else
            System.out.println("different memory location");
    }
}
class Employee implements Cloneable {
    int id;
    String name;
    String address;
    double salary;
    Employee(int id, String name, String address, double salary) {
        this.id = id;
        this.name = name;
        this.address = address;
        this.salary = salary;
    }
    @Override
    protected Object clone() throws CloneNotSupportedException {
        return super.clone();
    }
    @Override
    public String toString() {
        return "Employee{" + "id=" + id + ", name=" + name + ", address="
+ address + ", salary=" + salary + '}';
    }
}
```

Output:

```
PS D:\OOPS-PCC-CS593\Day-31-(17.02.2021)> javac ObjectCloningEx.java
PS D:\OOPS-PCC-CS593\Day-31-(17.02.2021)> java ObjectCloningEx
Employee{id=101, name=Rohit, address=Delhi, salary=95725.75}
different memory location
```