Correzione verifica 01/04/2023 Sirico Davide
Es1) Corretto
Es2) Corretto
Es3) Corretto
Es4) Corretto
Es5) Corretto
Es6) Sbagliato

```
import java.util.Queue;
import java.util.LinkedList;
import java.io.*;

public class Main
{
     public static void main(String[] args)
     {
          try {
               FileReader fileReader = new
FileReader("src/numeri.txt");
               BufferedReader input = new
BufferedReader(fileReader);

               String currentNumber;
               if((currentNumber=input.readLine())==null)
               {
                    System.out.println("File vuoto");
                    return;
               }

               Queue<Integer> C1 = new LinkedList<>();
               int number = Integer.parseInt(currentNumber);
               C1.add(number);
               while((currentNumber=input.readLine())!=null)
               {
                    number = Integer.parseInt(currentNumber);
                    C1.add(number);
               }
               stampa(C1);
               elimina(C1);
               stampa(C1);

          } catch(IOException e)
          {
               System.out.println("Impossibile leggere il file");
          } catch(NumberFormatException e)
          {
               System.out.println("Inserire solo numeri nel file");
          }
     }
     public static void stampa(Queue<Integer> C)
     {
          System.out.println("C1" + C);
     }
```

```
/*
public static void elimina(Queue<Integer> C)
{
     Queue<Integer> temp1 = new LinkedList<>();
     Queue<Integer> temp2 = new LinkedList<>();
     Queue<Integer> temp3 = new LinkedList<>();
     Queue<Integer> ris = new LinkedList<>();

     C = reverse(C);
     while(!C.isEmpty())
     {
          int current = C.remove();
          temp1.add(current);
          temp2.add(current);
     }
     temp1 = reverse(temp1);

     while(!temp1.isEmpty())
     {
          int top = myPeek(temp1);
          boolean duplicato = false;
          int current = temp1.remove();
          while(!temp2.isEmpty())
          {
               temp2.remove();
          }
          temp3 = copy(temp2);
          while(!temp2.isEmpty())
          {
               if(top==current)
               {
                    duplicato = true;
                    temp1.remove();
               }
               current = temp2.remove();
          }
          if(!duplicato)
          {
               ris.add(current);
          }
     }
     System.out.println("temp1: "+temp1);
     System.out.println("temp2: "+temp2);
     System.out.println("temp3: "+temp3);
     System.out.println("ris: "+ris);
}
public static int myPeek(Queue<Integer> C)
{
     Queue<Integer> temp = new LinkedList<>();

     int top = C.remove();
```

```
        temp.add(top);
        while(!C.isEmpty())
        {
            temp.add(C.remove());
        }

        while(!temp.isEmpty())
        {
            C.add(temp.remove());
        }

        return top;

    }
    public static Queue<Integer> reverse(Queue<Integer> Q)
    {
        if(Q.isEmpty())
        {
            return Q;
        } else{
            int current = Q.remove();
            Q = reverse(Q);
            Q.add(current);
            return Q;
        }
    }
    public static Queue<Integer> copy(Queue<Integer> Q)
    {
        Queue<Integer> temp1 = new LinkedList<>();
        Queue<Integer> temp2 = new LinkedList<>();
        while(!Q.isEmpty())
        {
            int current = Q.remove();
            temp1.add(current);
            temp2.add(current);
        }

        while(!temp2.isEmpty())
        {
            Q.add(temp2.remove());
        }
        return temp1;

    }
    */
        public static void elimina(Queue<Integer> C)
        {
            Queue<Integer> temp = new LinkedList<>();
            while(!C.isEmpty())
            {
                int current = C.remove();
                temp.add(current);
            }
```

```
            while(!temp.isEmpty())
            {
                  int current = temp.remove();
                  if(!isDuplicate(C, current))
                  {
                        C.add(current);
                  }
            }
      }
      public static boolean isDuplicate(Queue<Integer> C, int n)
      {
            Queue<Integer> temp = new LinkedList<>();
            boolean duplicate = false;
            while(!C.isEmpty())
            {
                  int current = C.remove();
                  if(current==n)
                  {
                        duplicate = true;
                  }
                  temp.add(current);
            }
            while(!temp.isEmpty())
            {
                  C.add(temp.remove());
            }
            return duplicate;
      }
}
```