```java
import java.util.ArrayList;
import java.util.LinkedList;


public class Main{
    public static void main(String[] args) {

        /*
        1 - Creare una LinkedList di codici alfabetici e numerici vuota, aggiungere i seguenti elementi:
        Pippo27
        Milena
        99
        e visualizzare la lista.
        */

        //Soluzione:
        // Creo una lista
        LinkedList ListaColori = new LinkedList();
        //Aggiungo vari elementi
        ListaColori.add("Pippo27");
        ListaColori.add("Milena");
        ListaColori.add(99);
        // Stampo la LinkedList
        System.out.println("Contenuto della LinkedList: " + ListaColori);



        /*
        2 - Creare una LinkedList di numeri interi vuota, aggiungere i seguenti elementi:
        10
        20
        30
        e visualizzare la lista.
        */

        //Soluzione:
        // Creo una lista di interi vuota
        LinkedList<Integer> ListaInteri = new LinkedList<>();
        // Aggiungo vari elementi
        ListaInteri.add(10);
        ListaInteri.add(20);
        ListaInteri.add(30);
        // Stampo la LinkedList
        System.out.println("Contenuto della LinkedList: " + ListaInteri);


        /*
        3 - Creare una LinkedList di stringhe vuota, aggiungere i seguenti elementi:
        Rosso
        Verde
        Giallo
        e visualizzare la lista  usando una iterazione.
        */

        //Soluzione:
        // Creo una lista di stringhe vuota
        LinkedList<String> Colori = new LinkedList<>();
        // Aggiungo vari elementi
        Colori.add("Rosso");
        Colori.add("Verde");
        Colori.add("Giallo");
        // Stampo la LinkedList
        System.out.println("Contenuto della LinkedList: " + Colori);
         // Stampo la LinkedList usando un loop
        System.out.println("Elementi LinkedList mediante loop:");
        for(String Colore: Colori)
            System.out.println (Colore);


        /*
        4 - Creare una LinkedList vuota e aggiungere gli elementi "D" e "G".
```

```
72          Aggiungere poi, mantenenendo l'ordine alfabetico della lista, l'elemento "A"
            e "M".
73          */
74
75          //Soluzione:
76          // Creo una lista vuota
77          LinkedList Lista = new LinkedList();
78
79          // Aggiungo vari elementi
80          Lista.add("D");
81          Lista.add("G");
82          Lista.addLast("M");
83          Lista.addFirst("A");
84          // Stampo la LinkedList
85          System.out.println("Contenuto della LinkedList : " + Lista);
86
87          /*
88          5 - Visualizzare l'index dell'elemento "G" dell'esercizio precedente
89          */
90
91          // get the index for "G"
92          System.out.println("Index for G:" + Lista.indexOf("G"));
93
94
95          /*
96          6 - Creare una ArrayList e aggiungere gli elementi "H" e "I". Aggiungere poi
            ArrayList alla LinkedList dell'esercizio precedente.
97          */
98
99          // Creo e inizializzo an ArrayList
100         ArrayList aLista = new ArrayList<>();
101         aLista.add("H");
102         aLista.add("I");
103
104         // aggiungo ArrayList a linkedList usando il metodo addAll
105         Lista.addAll(aLista);
106
107         // Stampo la LinkedList
108         System.out.println("Contenuto della LinkedList dopo aggiunta di ArrayList: " +
             Lista);
109
110
111         /*
112         7 - Da una LinkedList rimuovere l'elemento B, l'elemento in posizione 3, il
            primo e l'ultimo.
113         */
114
115         // Uso vari metodi per rimuovere elementi dalla linkedList
116         Lista.remove("B");
117         Lista.remove(3);
118         Lista.removeFirst();
119         Lista.removeLast();
120
121         // Stampo la LinkedList
122         System.out.println("Linked list dopo eliminazione: " + Lista);
123
124
125         /*
126         8 - Verificare se l'elemento G è contenuto in una LinkedList ed visualizzare
            il risultato.
127         */
128
129         // Uso il metodo contains per verificare se un elemento è in linkedList
130         boolean risultato = Lista.contains("G");
131
132         // Stampo ilrisultato
133         if(risultato)
134             System.out.println("La lista contiene l'elemento G");
135         else
136             System.out.println("La lista non contiene l'elemento G");
137
138
139         /*
```

```
        9 - Modificare l'elemento in posizione 2 con l'elemento J.
        */

        Lista.set(2, "J");
        System.out.println("LinkedList dopo modifica : " + Lista);

        /*
        10 - Data la class ListaConcatenata eliminare l'item 20.
        Scrivere poi un metodo della class ListaConcatenata che modifichi l'item in
        posizione 3
        */



        // Istanza della classe ListaConcatenata
        ListaConcatenata Elenco = new ListaConcatenata();

        //Dichiaro le variabili necessarie iniziali
        final int NMAX = 10;
        int[] info = new int[NMAX];
        int[] link = new int[NMAX];
        int item;

        //POPOLOAMENTO DI INFO E LINK fino a (DISP-1)
        int START = 1;
        int DISP = 6;
        info[0] = START;    // START lista dei nodi occupati
        link[0] = DISP;     // START lista dei nodi disponibili
        System.out.println("Popolamento nodi");
        int i;
        for(i=1; i<=info.length-(DISP-1); i++) {
            info[i] = i*100;
            link[i] = i+1;
        }
        // Identifico l'ultimo nodo con il fine lista -1
        link[info.length-(DISP-1)] = -1;

        //POPOLOAMENTO DI INFO E LINK da (DISP)
        for(i=DISP; i<info.length; i++) {
                info[i] = 0;
                link[i] = i+1;
        }
        // Identifico l'ultimo nodo con il fine lista -1
        link[info.length-1] = -1;
        System.out.println("");



        int loc = Elenco.RicercaLocOrd (info, link, 20);

        System.out.println("Eliminazione item 20");
        //loc = 3;
        int ris = Elenco.EliminaDopoLoc (info, link, loc);
        if (ris > 0)
            System.out.println("ELIMINATO DALLA LOCAZIONE " + ris);
        else
            System.out.println("ERRORE NON PREVISTO");
        System.out.println("");

    }
}

class ListaConcatenata {
    // METODI ATTRAVERSAMENTO ==============================

    // METODO ATTRAVERSAMENTO COME ARRAY
    public void Attraversamento (int[] info, int[] link) {
        System.out.println("i info link");
        for (int i=0; i<info.length; i++) {
            System.out.println(i + " " + info[i] + "  " + link[i]);
        }
    }

```

```java
    // METODO ATTRAVERSAMENTO INFO COME LISTA CONCATENATA
    public void AttraversamentoI (int[] info, int[] link) {
        int START   = info[0];
        int DISP    = link[0];      // NON OCCORRE
        int ptr     = START;
        System.out.println("i info link");
        while(ptr != -1) {
            System.out.println(ptr + " " + info[ptr] + "   " + link[ptr]);
            ptr = link[ptr];
        }
    }

    // METODO ATTRAVERSAMENTO DISP COME LISTA CONCATENATA
    public void AttraversamentoD (int[] info, int[] link) {
        int START   = info[0];      // NON OCCORRE
        int DISP    = info[0];
        int ptr     = link[0];
        System.out.println("i info link");
        while(ptr != -1) {
            System.out.println(ptr + " " + info[ptr] + "    " + link[ptr]);
            ptr = link[ptr];
        }
    }


    // METODO RICERCA LOC per ins IN INFO COME LISTA CONCATENATA ORDINATA
    static int RicercaLocOrd (int[] info, int[] link, int item) {
        int inizio_info = info[0];
        int inizio_disp = link[0];
        int ptr = inizio_info;
        int loc = 0;
        int SAVE = inizio_info; ptr = link[inizio_info];
        while(ptr != -1) {
            if (info[ptr] > item){
                loc = SAVE;
                break;
            }
            SAVE = ptr; ptr = link[ptr];
        }
        loc = SAVE;
        return loc;
    }

    // METODI INSERIMENTO ITEM ============================

    // METODO INSERIMENTO ITEM IN START LISTA CONCATENATA
    static int InserimentoInSTART (int[] info, int[] link, int item) {
        // Pre
        int START       = info[0];
        int DISP        = link[0];
        int risultato   = 0;
        int NEWnodo     = 0;

        // Controllo esistenza nodi disponibili
        if (DISP == -1) {
            // Nessuna disponibilita
            risultato = -1;
        }
        else {
            // Toglie il primo nodo da lista disponibili
            NEWnodo = DISP;
            DISP    = link[DISP];

            // Copia il nuovo dato in nuovo nodo
            info[NEWnodo] = item;

            // Aggiorna i puntatori
                // Il nuovo nodo punta ora al primo
                link[NEWnodo] = START;

                // Cambia START in modo che punti al nuovo nodo
                START = NEWnodo;

```

```
285             // Aggiorna START e DISP
286             info[0] = START;
287             link[0] = DISP;
288             // Visualizzazione solo per controllo
289             System.out.println("Nuovo START = " + START);
290             System.out.println("Nuovo DISP = "  + DISP);
291 
292             risultato = NEWnodo;
293         }
294         return risultato;
295     }
296 
297     // METODO INSERIMENTO ITEM IN LISTA CONCATENATA DOPO loc
298     static int InserimentoDopoLoc (int[] info, int[] link, int item, int loc) {
299         // Pre
300         int START       = info[0];
301         int DISP        = link[0];
302         int risultato   = 0;
303         int NEWnodo     = 0;
304 
305         // Controllo esistenza nodi disponibili
306         if (DISP == -1) {
307             // Nessuna disponibilita
308             risultato = -1;
309         }
310         else {
311             // Toglie il primo nodo da lista disponibili
312             NEWnodo = DISP;
313             DISP = link[DISP];
314 
315             // Copia il nuovo dato in nuovo nodo
316             info[NEWnodo] = item;
317 
318             //Inserisce dopo nodo con locazione loc.
319             // Aggiorna i puntatori
320                 // Il nuovo nodo punta ora al nodo puntato prima da loc
321                 link[NEWnodo] = link[loc];
322 
323                 // Cambiamo in modo che loc ora punti al nuovo nodo
324                 link[loc] = NEWnodo;
325 
326             // Aggiorna START e DISP
327             info[0] = START;
328             link[0] = DISP;
329 
330             risultato   = NEWnodo;
331         }
332         return risultato;
333     }
334 
335 
336     // METODO ELIMINA ITEM DOPO LOC IN LISTA CONCATENATA
337     static int EliminaDopoLoc (int[] info, int[] link, int loc) {
338         // Pre
339         int START       = info[0];
340         int DISP        = link[0];
341         int risultato   = 0;
342         int ELInodo     = 0;
343 
344         if (loc == -1) {
345             // Si tratta del primo nodo
346             START = link[START];
347             ELInodo = START;
348             risultato = ELInodo;
349         }
350         else {
351             // Si tratta di nodo diverso dal primo
352             // Aggiorna i puntatori
353                 // Il nodo da eliminare è quello a cui punta loc
354                 ELInodo = link[loc];
355 
356                 // Cambiamo in modo che loc ora punti al link del nodo eliminato
357                 link[loc] = link[ELInodo];
```

```
358
359             risultato = ELInodo;
360         }
361
362         //Rende il nodo DISP
363         DISP = ELInodo;
364         link[ELInodo] = DISP;
365
366         // Aggiorna START e DISP
367         info[0] = START;
368         link[0] = DISP;
369
370         return risultato;
371     }
372 }
373
374
375 OUTPUT:
376 Contenuto della LinkedList: [Pippo27, Milena, 99]
377 Contenuto della LinkedList: [10, 20, 30]
378 Contenuto della LinkedList: [Rosso, Verde, Giallo]
379 Elementi LinkedList mediante loop:
380 Rosso
381 Verde
382 Giallo
383 Contenuto della LinkedList : [A, D, G, M]
384 Index for G:2
385 Contenuto della LinkedList dopo aggiunta di ArrayList: [A, D, G, M, H, I]
386 Linked list dopo eliminazione: [D, G, H]
387 La lista contiene l'elemento G
388 LinkedList dopo modifica : [D, G, J]
389 Popolamento nodi
390
391 Eliminazione item 20
392 ELIMINATO DALLA LOCAZIONE 2
393
394
```