Il presente compito va elaborato su foglio protocollo e consegnato prima della fine dell'ora.
Il punti indicati per ogni esercizio saranno assegnati se lo stesso è svolto in modo completo, corretto e ordinato in ogni sua parte.

Verifica n. 2 di INFORMATICA del secondo trimestre.

12-02-2025 – Classe: 4D – Alunno: _____

01 – (1punto) Creare una LinkedList di codici (per es. ListaCodici) alfabetici e numerici vuota, aggiungere i seguenti elementi: Pippo27; Milena; 99;
e visualizzare la lista. **(Facoltativo se prescritto maggiore tempo)**

02 - (1punto) Creare una LinkedList di numeri interi vuota (per es. ListaInteri), aggiungere i seguenti elementi: 10; 20; 30;
e visualizzare la lista.

03 - (1punto) Creare una LinkedList di stringhe vuota (per es. ListaColori), aggiungere i seguenti elementi: Rosso; Verde; Giallo;
e visualizzare la lista usando una iterazione.

04 - (1punto) Creare una LinkedList vuota (per es. Lista) e aggiungere gli elementi "D" e "G". Aggiungere poi, mantenendo l'ordine alfabetico della lista, l'elemento "A" e "M".

05 - (1punto) Visualizzare l'index dell'elemento "G" dell'esercizio precedente.

06 - (1punto) Creare una ArrayList (per es. aLista) e aggiungere gli elementi "H" e "I". Aggiungere poi ArrayList alla LinkedList dell'esercizio precedente.

07 - (1punto) Da una LinkedList (per es. Lista) rimuovere l'elemento B, l'elemento in posizione 3, il primo e l'ultimo.

08 - (1punto) Verificare se l'elemento G è contenuto in una LinkedList (per es. Lista) e visualizzare il risultato.

09 - (1punto) Modificare in una LinkedList (per es. Lista) l'elemento in loc 2 con l'elemento J.

10 - (1punto) Data la class ListaConcatenata elaborata in classe eliminare l'item 20.
Scrivere poi un metodo della class ListaConcatenata che modifichi l'item in posizione data.

```
class ListaConcatenata {
        // METODO ATTRAVERSAMENTO COME ARRAY
        public void Attraversamento (int[] info, int[] link) {}
        // METODO ATTRAVERSAMENTO INFO COME LISTA CONCATENATA
        public void AttraversamentoI (int[] info, int[] link) {}
        // METODO ATTRAVERSAMENTO DISP COME LISTA CONCATENATA
        public void AttraversamentoD (int[] info, int[] link) {}
        // METODO RICERCA ITEM IN INFO COME LISTA CONCATENATA
        static int Ricerca (int[] info, int[] link, int item) {}
        // METODO RICERCA ITEM IN INFO COME LISTA CONCATENATA ORDINATA
        static int RicercaOrd (int[] info, int[] link, int item) {}
        // METODO RICERCA LOC per ins IN INFO COME LISTA CONCATENATA ORDINATA
        static int RicercaLocOrd (int[] info, int[] link, int item) {}
        // RICERCA LOC per eliminare nodo IN INFO COME LISTA CONCATENATA ORDINATA
        static int RicercaEliOrd (int[] info, int[] link, int item) {}
        // METODO INSERIMENTO ITEM IN LISTA CONCATENATA DOPO loc
        static int Inserimento (int[] info, int[] link, int item, int loc) {}
        // METODO ELIMINA ITEM DOPO LOC IN LISTA CONCATENATA
        static int EliminaDopoLoc (int[] info, int[] link, int loc) {}
}
```

Il presente compito va elaborato su foglio protocollo e consegnato prima della fine dell'ora.
Il punti indicati per ogni esercizio saranno assegnati se lo stesso è svolto in modo completo, corretto e ordinato in ogni sua parte.

Modifier and Type	Method and Description
boolean	**add**(E e) Appends the specified element to the end of this list.
void	**add**(int index, E element) Inserts the specified element at the specified position in this list.
boolean	**addAll**(Collection<? extends E> c) Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's iterator.
boolean	**addAll**(int index, Collection<? extends E> c) Inserts all of the elements in the specified collection into this list, starting at the specified position.
void	**addFirst**(E e) Inserts the specified element at the beginning of this list.
void	**addLast**(E e) Appends the specified element to the end of this list.
void	**clear**() Removes all of the elements from this list.
boolean	**contains**(Object o) Returns true if this list contains the specified element.
E	**get**(int index) Returns the element at the specified position in this list.
E	**getFirst**() Returns the first element in this list.
E	**getLast**() Returns the last element in this list.
int	**indexOf**(Object o) Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
int	**lastIndexOf**(Object o) Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.
E	**remove**() Retrieves and removes the head (first element) of this list.
E	**remove**(int index) Removes the element at the specified position in this list.
E	**removeFirst**() Removes and returns the first element from this list.
E	**removeLast**() Removes and returns the last element from this list.
E	**set**(int index, E element) Replaces the element at the specified position in this list.
int	**size**() Returns the number of elements in this list.