Analisi:

# lezione 1: Liste concatenate

## lavoro:

Creare un array di Struct contenente ognuna la posizione del valore successivo e un contenuto come un nome sotto forma di stringa.  
In questo modo si ottiene una lista.  
I primi 10 valori sono assegnati da codice, gli altri 10 per esempio sono spazi vuoti pronti per essere riempiti.  
Inizializziamo 10 valori, che per semplicità usiamo 10 compagni di classe a caso.

## Opzioni menù:

* Inserimento.
* Rimozione.
* Visualizzazione Fisica.
* Visualizzazione Logica.
* Ricerca Valore.

## Struttura struct:

Chiamo la Struct “dato”.  
Chiamo il vettore di Struct “dati”.

La struttura delle struct sarà la seguente:

* Contenuto.
* Pos\_Suc.

Lo scopo del contenuto è quello di immagazzinare un valore sotto forma di stringa, in questo caso il nome o cognome del compagno di classe.

Lo scopo della posizione successiva è quello di puntare alla posizione in cui si trova il nome successivo nell’alfabeto.

## Inizializzazione:

Inserisco tramite codice i valori nel vettore di Struct, con le posizioni prefissate.

## Inserimento:

Verifico subito prima di eseguire l’inserimento se è disponibile qualche spazio vuoto, in caso affermativo procedo.  
Verifico se il valore da inserire viene prima nell’alfabeto del primo della lista, e nel caso sia minore modifico il puntatore globale del valore minore perché punti a quello inserito, e punto sempre quello appena inserito al vecchio minore.

In caso invece non sia il valore minore, tramite un ciclo verifico chi dovrà puntare al nuovo valore e chi dovrà puntare il nuovo valore, il ciclo continua fino a quando non sono stati letti tutti i valori tramite lettura logica oppure non venga trovato uno maggiore di quello inserito.  
Una volta trovati questi valori, verranno impostati ai valori necessari per mantenere l’ordine corretto.

## Rimozione:

Verifico subito una condizione di criticità, ossia se il nome da rimuovere inserito è il primo valore della lista, in caso affermativo ottengo la sua posizione puntata e la setto come nuovo valore minimo globale da cui iniziare la lettura logica (la posizione del primo elemento in sintesi) e poi setto ai valori di default “nulli” il valore alla posizione da rimuovere.

In caso normale invece, in cui l’elemento da rimuovere non sia il primo, tramite un ciclo while che continua fino a quando viene il valore viene trovato oppure la posizione puntata è quella nulla di riferimento (attualmente -1), cerco il valore leggendo la posizione di quello attuale in lettura logica e cercando con questa il valore del successivo, se corrisponde a quello da cercare, eseguo le azioni di modifica del puntatore del valore del ciclo attuale con quello del puntatore del ciclo “successivo” letto prima, e setto ai valori nulli di default il valore da rimuovere.

Il valore rimosso punterà a -2 che è un valore bandiera.

## visualizzazione fisica:

Principalmente utilizzato per motivi di debug, legge tutti i valori dal primo all’ultimo, senza un ordine logico ma fisico, ossia chi prima arriva e prima sarà letto.

## Visualizzazione logica:

Devo visualizzare i valori in base alla posizione a cui ognuno punta, fino a quando non raggiungo un valore che punta al valore bandiera nullo, al quale punto si ferma.

## Ricerca valore:

Chiedo all’utente di inserire il nome dell’utente e tramite un ciclo usando le posizioni a cui ogni elemento punta, cerco se la stringa del nome o cognome è contenuta, in caso positivo procedo comunico all’utente in che posizione logica è stato trovato, nel caso non venga trovato comunico all’utente che il valore inserito non è contenuto nel vettore di Struct.

# Parte 2:

## Rimuovere valori e recuperarli:

Leggo il vettore e trovo i valori con valore bandiera -2 come puntatore che significa che sono stati eliminati; infatti, nella visualizzazione logica nessuno punta a loro e di conseguenza non vengono mai letti, se voglio recuperarli devo leggere il loro valore e determinare che dovrebbe nelle nuove condizioni puntare a loro e chi loro dovrebbero puntare.

// Parole del prof:

Ci sono fondamentalmente 2 liste logiche, come se noi all’interno della classe avessimo 2 ordinamenti, una lista che punta ai suoi corretti, e stessa cosa nella seconda, e quello da fare sarebbe inserire i valori di una lista nell’altra nelle posizioni corretti, inizialmente PRIMA di ELIMINARE dei valori, una lista non contiene nulla, l’altra contiene i valori e quando saranno eliminati, l’altra conterrà dei valori eliminati.

Di fatto abbiamo due liste, possiamo intendere una lista libera di elementi non usati (LL), e una lista normale di elementi (LC) che contiene tutti gli elementi ordinati che si puntano. Manca roba qui.