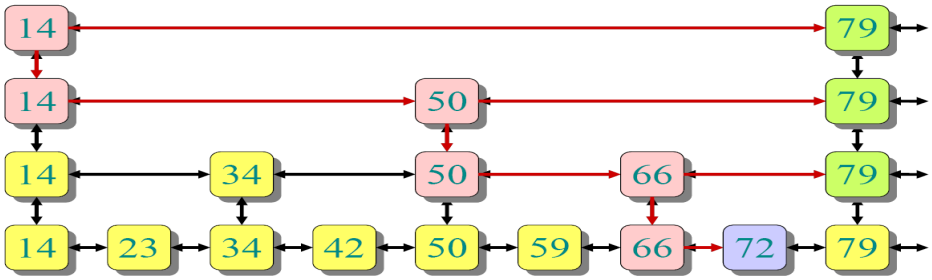
Cognome Nome matricola

Skip list (adapted from Wikipedia, the free encyclopedia)

A skip list is a probabilistic data structure, based on multiple parallel, sorted linked lists, with efficiency comparable to a binary search tree. ... Insert, search and delete operations are performed in logarithmic randomized time. ...
Description
A skip list is built in layers. The bottom layer is an ordinary ordered linked list. Each higher layer acts as an "express lane" for the lists below, where an element in layer (*i*) appears in layer *(i+1)* with some fixed probability p (two commonly-used values for p are 1/2 or 1/4).
A search for a target element begins at the head element in the top list, and proceeds horizontally until the current element is greater than or equal to the target. If the current element is equal to the target, it has been found. If the current element is greater than the target, the procedure is repeated after returning to the previous element and dropping down vertically to the next lower list. The expected number of steps in each linked list is 1/p, which can be seen by tracing the search path backwards from the target until reaching an element that appears in the next higher list. Therefore, the total expected cost of a search is $(\log_{1/p} n)/p$, which is $\mathcal{O}(\log n)$ when p is a constant. By choosing different values of p, it is possible to trade search costs against storage costs. ...

– Example: Search for 72

- Level 1: 14 too small, 79 too big; go down 14
- Level 2: 14 too small, 50 too small, 79 too big; go down 50
- Level 3: 50 too small, 66 too small, 79 too big; go down 66
- Level 4: 66 too small, 72 spot on



Punti 8/30 - Realizzare uno schema UML delle classi che permetta la realizzazione di un sistema informativo basato su *indici* di ricerca di tipo *SkipList.* Considerando che le informazioni delle *SkipList* costituiscono le chiavi identificative di informazioni, la lista (quella inferiore), che contiene tutte le chiavi, qualificherà anche le informazioni di tipo *Tdato*.
Punti 5/30 – Realizzare poi uno schema UML delle attività che permetta la ricerca di una informazione qualificata dalla sua chiave associata, sfruttando un opportuno iteratore.

Cognome Nome matricola

Punti 5/30 – **UML Talking State Machines**

For this task, you have to build two state machines that communicate with each other via a simple query/response protocol. One state machine is a **Transmitter**, and the other one is a **Receiver**. The state machines use a protocol that uses two control lines: *TransmitReady* and *DataRequest,* and a set of n data lines. The procedure is as follows:

- **Receiver** tests the *TransmitReady* line to see if it is not asserted. It repeats this step until the *TransmitReady* line is not asserted;
- if *TransmitReady* is not asserted, **Receiver** requests a transmission of data by asserting the *DataRequest* line;
- when **Transmitter** sees the *DataRequest* line asserted, it places the current data on the data lines and then asserts *TransmitReady;*
- when **Receiver** sees the *TransmitReady* line asserted, it reads the data on the data lines (Receive State) and then de-asserts *DataRequest;*
- when **Transmitter** sees the *DataRequest* line de-asserted, it de-asserts the *TransmitReady* line.

Note:
Tutti i diagrammi disegnati devono utilizzare la sintassi del linguaggio UML ed essere **opportunamente commentati**.
Utilizzare il foglio protocollo per la brutta copia e gli spazi liberi delle pagine 1 e 2 del presente testo, per disegnare i diagrammi definitivi utilizzando la penna ed inoltre scrivere il proprio nome sui tutti i fogli. Qualora si utilizzasse anche il foglio protocollo per i diagrammi definitivi, indicarlo nel presente foglio.

Cognome Nome matricola

Punti 4/30

Punti 4/30

Punti 4/30